Expressive Movement Generation with Machine Learning

by

Omid Alemi

M.Sc., University of Northern British Columbia, 2012
B.Sc., University of Arak, 2009

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Interactive Arts + Technology
Faculty of Communication, Art and Technology

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Omid Alemi**
Degree:	**Doctor of Philosophy**
Thesis title:	**Expressive Movement Generation with Machine Learning**
Committee:	**Chair:** Halil Erhan Associate Professor, School of Interactive Arts & Technology

Philippe Pasquier
Supervisor
Associate Professor, School of Interactive Arts & Technology

Thecla Schiphorst
Committee Member
Professor, School of Interactive Arts & Technology

Steve DiPaola
Committee Member
Professor, School of Interactive Arts & Technology

Baptiste Caramiaux
Examiner
Research Scientist
Sorbonne

Michiel van de Panne
External Examiner
Professor
Department of Computer Science
University of British Columbia

Ethics Statement

The author, whose name appears on the title page of this work, has obtained, for the research described in this work, either:

a. human research ethics approval from the Simon Fraser University Office of Research Ethics

or

b. advance approval of the animal care protocol from the University Animal Care Committee of Simon Fraser University

or has conducted the research

c. as a co-investigator, collaborator, or research assistant in a research project approved in advance.

A copy of the approval letter has been filed with the Theses Office of the University Library at the time of submission of this thesis or project.

The original application for approval and letter of approval are filed with the relevant offices. Inquiries may be directed to those authorities.

Simon Fraser University Library
Burnaby, British Columbia, Canada

Update Spring 2016

Abstract

Movement is an essential aspect of our lives. Not only do we move to interact with our physical environment, but we also express ourselves and communicate with others through our movements. In an increasingly computerized world where various technologies and devices surround us, our movements are essential parts of our interaction with and consumption of computational devices and artifacts. In this context, incorporating an understanding of our movements within the design of the technologies surrounding us can significantly improve our daily experiences. This need has given rise to the field of movement computing – developing computational models of movement that can perceive, manipulate, and generate movements.

In this thesis, we contribute to the field of movement computing by building machine-learning-based solutions for automatic movement generation. In particular, we focus on using machine learning techniques and motion capture data to create controllable, generative movement models. We also contribute to the field by creating datasets, tools, and libraries that we have developed during our research.

We start our research by reviewing the works on building automatic movement generation systems using machine learning techniques and motion capture data. Our review covers background topics such as high-level movement characterization, training data, features representation, machine learning models, and evaluation methods. Building on our literature review, we present WalkNet, an interactive agent walking movement controller based on neural networks. The expressivity of virtual, animated agents plays an essential role in their believability. Therefore, WalkNet integrates controlling the expressive qualities of movement with the goal-oriented behaviour of an animated virtual agent. It allows us to control the generation based on the valence and arousal levels of affect, the movement's walking direction, and the mover's movement signature in real-time. Following WalkNet, we look at controlling movement generation using more complex stimuli such as music represented by audio signals (i.e., non-symbolic music). Music-driven dance generation involves a highly non-linear mapping between temporally dense stimuli (i.e., the audio signal) and movements, which renders a more challenging modelling movement problem. To this end, we present GrooveNet, a real- time machine learning model for music-driven dance generation.

Keywords: human movement; movement computing; machine learning; generative systems; computational creativity; motion capture

Dedication

To my parents.

Acknowledgements

I would like to thank my supervisor, Dr. Philippe Pasquier, for his valuable direction, mentorship, and guidance. I could not have completed my PhD studies without his encouragement and patience. I also would like to thank my committee members, Dr. Thecla Schiphorst and Dr. Steve DiPaola, for their support throughout my PhD studies. I also would like to acknowledge my colleagues and friends from the Moving Stories project and the Metacreation Lab, William Li, Jules Françoise, Jianyu Fan, Ulysses Bernardet, Shannon Cuykendall, Sarah Fdili-Alaou, Karen Cochrane, and Kristin Carlson.

I wish to thank my parents and family. I would not be able to start and finish my studies without the selfless support of my parents throughout my life and in particular during the long years of my PhD studies. I am deeply grateful for their unconditional love.

This research was funded by the Social Sciences and Humanities Research Council and the Natural Sciences and Engineering Research Council of Canada.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction and Background

Movement is an essential aspect of human life. We perform our daily tasks, interact with objects, communicate with other individuals, and express ourselves, to a large extent, through our movements. Our movements also influence how we interact with and consume technology. With the trend in recognizing the role of movement in cognition (Wilson and Golonka, 2013) and in human-computer interaction research (Kirsh, 2013; Bernardet et al., 2016; Candau et al., 2017), more attention is being paid to incorporate movement understanding in technologies. Consequently, a growing body of research targets fundamental and applied aspects of building computational models of human movement in technologies. There are three types of problems that research on computational human movement models deals with: movement perception, movement manipulation, and movement generation. We describe each in the following.

Movement perception studies how one interprets the movements of a mover. Movement is a form of non-verbal communication and conveys information about the mover (Troje, 2002). For example, research has shown that the emotional state can be perceived from a person's movement (Clarke et al., 2005; Li and Pasquier, 2016; Fourati and Pelachaud, 2018; Li et al., 2018). Therefore, movement perception allows humans to orient themselves with respect to their surrounding environment and other individuals (Troje, 2008). Consequently, inferring such information from movement allows computational systems to better understand and interact with humans. For example, a model can infer the action, predict the hand's target position, or estimate the mover's affective state from movement.

There is a growing market for including movement perception in technologies and consumer electronic products. For example, movement perception is being used for problems such as gesture detection when interacting with a device (e.g., mobile phones), detection of full-body movements while playing a video game (e.g., via Microsoft Kinect or virtual reality headsets), and movement analysis in health applications. More recently, products such as *Project Soli* from Google provide general-purpose movement sensing platforms for building movement-aware consumer products (Wang et al., 2016; Google ATAP, 2020).

Research on human movement manipulation looks at ways to modulate movement data so that the way human observers perceive it in animation changes to a desired state. Instead of recording motion capture data or creating new animations from scratch, manipulation techniques allow augmenting existing animation data to create the desired animated content. For example, given one or more segments of motion capture data for picking up an object, how can one change modify the data for the hand to follow a given trajectory while keeping the resulting movement realistic.

Movement generation research aims to build systems that can synthesize movement data that are naturally-looking and believable. We can then use the generated movement data to animate a software agent or control a physical robot. Movement generation models can

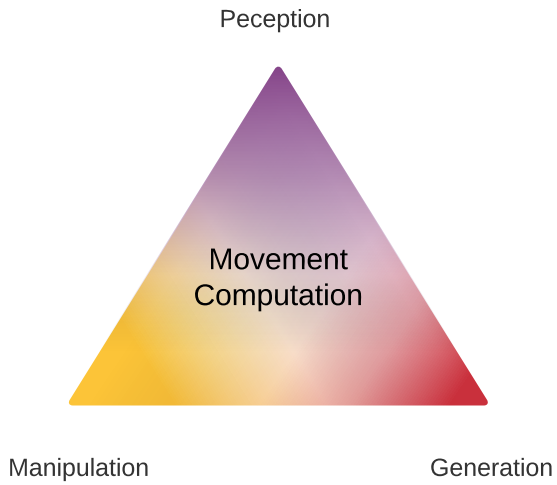


Figure 1.1: Three aspects of human movement computation

be used by animators to synthesize new animations. They can also be used as movement controllers for video games. A model that can generate movement dynamics can also be used to derive the movements of non-anthropomorphic objects, animated or physical. For example, *R* is a dancing speaker from Teenage Engineering (Teenage Engineering, 2020) which moves its robotic arm based on the music that it is playing.

Note that these three groups of problems are often intertwined and working on one often requires addressing others as well (Figure 1.1). For example, affective computing research on movement generation considers problems such as how the mover's affective state is perceived and how to generate or manipulate movement data to change the perception while keeping the movement look realistic to human observers. Despite such interconnected interactions, and although movement perception, manipulation, and generation have been actively researched, there are no agreed-upon foundations for a broader field that encompasses them. Human movement computing, as this broader research field, is still under-developed and in its infancy. Unlike computer vision, natural language processing, and music, the field of human movement computing lacks comprehensive tools and libraries that provide a wide range of feature extraction algorithms, tests, metrics, and evaluation frameworks. While elements of this thesis contribute to the field, laying out the foundations of the field is beyond the scope of this thesis. We make the case for further collective work on building a stronger human movement computing community.

Creating computational models of human movement, and in particular movement manipulation and generation models, is a challenging task. The mover's internal state, such as her goals, plans, and affective state, influence her movements. Also, the body's physical characteristics (e.g., shape and weight) and the physical environment surrounding the body, both subject to physics laws, influence the movement. Furthermore, there are subtle variations in movements of different movers performing the same task. To address the challenges

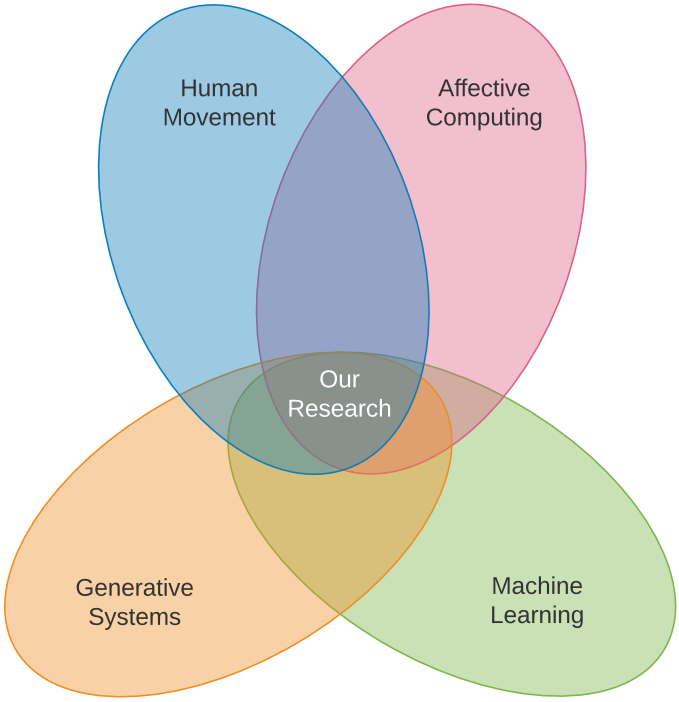


Figure 1.2: The focus of this thesis

we mentioned, we follow a data-driven approach and use machine learning techniques to learn the underlying generative patterns behind the movement from how humans perform them and generate and control new samples. In the next section, we lay out our focus within the field of human movement computing and describe the motivations behind our research.

1.2 Thesis Focus and Motivations

The overarching focus of this thesis is on human movement computing. Within the human movement computing framework, we particularly focus on movement generation, but also consider problems of perception and manipulation. We present research at the intersection of generative systems, machine learning, human movement, and affective computing (Figure 1.2). More specifically, our focus is on using machine learning approaches to develop movement generation models. Such models are built by training them on recorded movements of real human actors. As a result of the training process, the model learns the distribution of the represented group of movements in the training set. We can then use the model to draw new samples that look similar to those in the training set by an unbiased observer, a process also known as *style imitation*. Note that throughout this thesis, when talking about movement computing and movement generation, we refer to the representation of human movement through motion capture data and skeletal animation of virtual humanoid characters.

For movement generation models, the ability to control the generation is essential for many applications in which the movement has to satisfy a specific set of qualities or criteria. To this end, we have to address problems such as what qualities and characteristics to use for control, how to *represent* them, and how to use the given control representation to *control* the movements that the model generates. As we were starting our research, our review of the literature showed that movement generation models in the literature either did not address the control problem or they merely controlled arbitrary qualities such as walking in a chicken or drunken style (Section 2.3). Therefore, we aimed to use meaningful descriptions for controlling the generation. To characterize such meaningful descriptions, we devise a multi-layer framework to represent the movement's high-level characteristics. We chose the layers to correspond to the mover's internal state, such as its affective state, goals, plans, and the mover's movement signature (Section 2.3.2). We follow this framework in designing our movement controllers in Chapters 5 and 6. We also experiment with low-level and multi-modal data such as raw audio to control the movement generation, as presented in Chapters 7 and 8.

Our motivations in building generative movement models are threefold. First, we are motivated by scientific research since we can build better computational models of movement by building generative models that better understand movements. Second, we are motivated by the application of automatic movement generation because of the need for more movement content. Third, we pursue artistic goals as we are interested in creating computational means to create artwork involving human movement. We elaborate on each motivation in the following.

By building generative models that predict how a movement progresses through time, we essentially create models that can process human movement and "understand", to a reasonable extent, the qualities associated with the movements. Such models, in effect, learn the distribution of the movements they are trained on. As a result, a generative model of movement has applications beyond merely generating new movement animation. By being able to predict movements, we can design and develop better approaches to incorporate movement and body awareness in new technologies. Further, a generative model might perform better in discriminating between different movement qualities or filling in missing or noisy data compared to non-generative approaches.

The works presented here are also motivated by the need for more movement data in the animation and video game industries. Creating human movement animation from scratch is a challenging task for animators. As a result, the animation industry uses motion capture technologies to record real human actors' performance, represented in the form of position or orientation of several body joints. Animators can then use this numerical representation of movement as the basis to create realistically-looking movement animation. Yet, this process is costly and time-consuming and does not scale well with the increasing demand for more content, including character animation, from non-linear media such as

video games. Moreover, it is not feasible to capture every variation of every movement for a particular scenario. These issues, among other things, pushed us to look for ways to create models that can automatically generate movement animations with the desired qualities. A generative model can help as both a movement controller and a computer-assisted creativity tool by the animators. In particular, we focus on the affective qualities of movement because such affective control increases the generated movement's believability. Since the model generates affective variations of movements on demand, it can be used as a computer-assisted creativity tool by the animators.

Finally, we are motivated to develop computational means that allows artists to create artistic expressions of movement. Movement generation models not only allow an artist to create interactive animation content; they also provide the means to design movement-inspired motions for products and designs.

Given this thesis's aforementioned focus on movement generation and controlling the movement qualities, we developed the research questions presented in the following section.

1.3 Research Questions and Contributions

The research presented in this thesis is formed and developed around four research questions. While these questions guided our research, our findings also helped us develop and further refine the questions and our direction during our research process. The four questions are as follows::

RQ1: **What is the state of the art on using machine learning techniques and motion capture data to create movement generation systems?**

As we started our research on movement generation models, we realized that the research community lacked a comprehensive review of the fast-evolving field of machine-learning-based movement generation. There were review papers that covered other topics in the broad field of automatic movement generation. Most notably, the reviews covered the field of automatic movement generation in general (Geijtenbeek and Pronost, 2012), the physics-based approaches to modelling movement (Wang et al., 2014), and data-driven approaches to generate and manipulate movement data (Pejsa and Pandzic, 2010; Karg et al., 2013). However, no work focused on the implications of training machine learning models on motion capture data to develop generative systems. As a result, we decided to write a review paper with such focus.

Our Contributions: This thesis contributes to RQ1 by conducting an extensive review of the literature and providing a comprehensive resource on training machine learning techniques on motion capture data. We summarize and present the advantages and limitations of key machine learning models and possible solutions to each model's common problems. In addition, we identify the key goals, challenges, and gaps in the field. We discuss differ-

ent representations of movement data and pre-processing methods. Finally, we elaborate on using different machine learning techniques with motion capture data for learning and generating movement animation.

RQ2: **What tools and data do we need to support the research on machine learning and motion capture data?**

Parallel to the gaps in the field outlined in RQ 1, we noticed the shortcomings in both the tools and the data to support machine learning research on motion capture data. Most libraries for processing motion capture data were not compatible with the modern machine learning pipelines and deep learning libraries. There was also a lack of tools to easily visualize, animate, or analyze movement data without resorting to using 3D animation software. As a result, we set our goal to create such tools and libraries. Furthermore, there were no publicly available motion capture datasets to use for the particular research questions we sought after. Namely, we needed a dataset that contains movements with variations in their affective state based on a dimensional affect representation for RQ 3 and a dataset of synchronized dance and music recordings for RQ 4. As a result, we curated and captured such datasets.

Our Contributions: We developed open-source tools, libraries, and models tailored towards supporting human movement computation and machine learning research on motion capture data. Namely:

- *MoDa UI*: A web-based front-end for easy access to the data.
- *Movement Data*: Two published high-quality motion capture datasets: the *Affect-Expressive Movements* and the *GrooveDB*.
- *Mova*: a web-based movement analytics platform (Alemi et al., 2014).
- *PyMO*: a Python library that provides various tools to process and analyze motion capture data tailed for machine learning applications.
- *AffectRank*: a machine learning model for estimation of affect rank from motion capture data (Li et al., 2018).
- m+m: a framework for building real-time and interactive systems that use movement data (Bernardet et al., 2016).

RQ3: **Can we do *style imitation* for movement generation? And if so, what characteristics of the generated movements can we *control*?**

Style imitation is the process of generating new samples that are deemed similar to a set of existing samples by an unbiased observer. For example, given a set of paintings with the style of van Gogh, create new paintings are classified as the style of van Gogh. Through

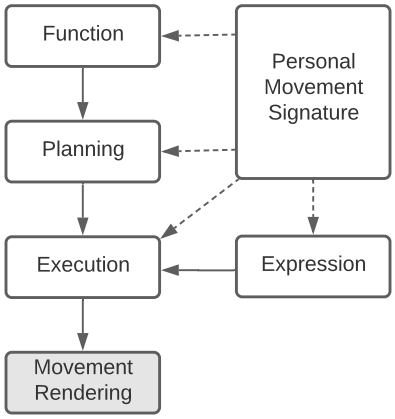


Figure 1.3: The multi-layer framework to describe the movement of virtual agents. Solid arrows represent direct influence of one dimension on another while dashed arrows represent indirect influences.

exploring RQ 1, we identified different approaches introduced in the literature to perform style imitation on movement data. However, most works did not support mechanisms for controlling the characteristics of their output. Among those that provided a control mechanism, most lacked a semantically meaningful relationship between the control and virtual agents. In parallel, following our group's research that show humans can perceive the affective qualities from the motion capture data to a reasonable extent (Li and Pasquier, 2016; Li et al., 2018), we asked the question if we can also control and generate such affective qualities.

Coming from a multi-agent systems background, we look at the problem of control in movement generation from a virtual agent perspective. More clearly, we are interested in creating movement generation models that are linked to the internal state of a moving agent (e.g., a human, a robot, or an animated avatar). Within this perspective, controlling the movement generation is in response to the changes in the internal state of the mover, such as its goals, beliefs, plans, or affective state. To better understand and formulate different aspects of agent movement control, we devised a multi-layer framework for characterizing the movements of moving agents, tailored for controlling the movement generation. The framework, as shown in Figure 1.3, describes movement across the dimensions of function, planning, execution, expression, personal movement signature, and movement rendering. We elaborate on each dimension in Section 2.3.2. We note that the proposed framework is not intended as a general framework for describing human movement, but as a tool to formulate our research in movement control and generation for virtual agents. For RQ 3, we follow this framework to control movement across three dimensions of the proposed framework: the *planning*, the *expressive*, and the *personal movement signature*. Accordingly, we formulated the problem into three sub-questions:

- **RQ 3.1.** Can we control the perceived affective qualities during the generation?
- **RQ 3.2.** Can we control the personal variations (personal movement signature) in the movement?
- **RQ 3.3.** Can we control the planning aspects of movement?

Our Contributions: Our research shows that we can train machine learning models to generate movements while controlling the movement's perceived affective qualities. Our models can also generalize over the space of affective qualities. While we train the model on data containing samples from only nine affective states, it can generate samples representing affective qualities beyond those in the training set. Ultimately, we present a neural-network-based movement controller that allows for real-time generation and control of affect expression, the personal movement signature of the mover, and movement planning in the form of steering the walking movements.

RQ4: **Can we teach a machine learning model to generate dancing movements to a streaming audio track?**

Once we were able to control the movement generation using high-level, temporally coarse stimuli such as the affective qualities, we decided to pursue designing models that control the movement generation using more complex stimuli such as music. Music-driven dance generation involves a highly non-linear mapping between temporally dense stimuli (i.e., the audio signal) and movements, which renders a more challenging problem in modelling movement. Addressing this problem has applications in the arts and video games and helps us develop better multi-modal machine learning applications.

Our Contributions: Our research contributes to RQ 4 by presenting *GrooveNet*, a real-time machine learning model for music-driven dance generation. In addition, we provide a publicly available data set of synchronized music and dance that we captured to develop GrooveNet.

1.4 Structure of the Thesis

1.4.1 How to read this thesis

This thesis contains the cumulation of multiple works, published or yet-to-be-published, presented in separate chapters (refer to Appendix A for more details on the guidelines for cumulative theses). This introduction chapter outlines the overarching scope and motivation, the research questions, and the works' contribution presented in the thesis. This chapter also serves as a guide to how to read and navigate the rest of the thesis. The overall thesis structure is shown in Tables 1.1 and 1.2. The tables indicate the contributions of each

chapter and appendix to the research questions. For chapters and appendices that represent a published paper, the publication column corresponding to the publication numbers outlined in Section 1.4.3. Further, we outline each chapter's summary and their relationship to the research questions, contributions, publications, and other chapters and appendices in the following.

Note on Iterative Works: Because of the independent nature of some of the chapters in the thesis (i.e., self-contained publications), some content within the overall thesis can be repetitive. The publications presented in Chapters 5 and 6 are iterations of the same work. Therefore, Chapter 6 can be read fully to understand the final work while Chapter 5 can be read selectively as a reference to the first iteration of the work. Similarly, the publications presented in Chapters 7 and 8 are iterations of the same work. Although they share a similar review of the literature and the description of the problem, the two chapters differ on the content of the training set, the representation of the data, and the machine learning techniques used.

Note on Collaboration and Contributions: Because of the cumulative nature of this thesis, each chapter presents a collaborative publication. Therefore, we explain the contributions of each collaborator per chapter.

Table 1.1: Organization of the thesis outlining the chapters and their contributions, as well as their relationship to the research questions. Chapters that represent a published paper are indicated by the corresponding publication number. The publication numbers refer to the list presented in Section 1.4.3. "–" indicates the content that has not been previously published.

Chapter	Contributions	Publication
1. Introduction	General Introduction, Research Questions, and Contribution	—
2. Machine Learning for Data-Driven Movement Generation: a Review of the State of the Art	Review of the state of the art on using machine learning techniques and motion capture data for movement generation (**RQ 1**)	**P 1**
3. Data and Tools for Human Movement Computing	• MoDa movement hub • The *Affect-Expressive Movements* and *GrooveDB* datasets • PyMO: A library for machine learning research with motion capture data • AffectRank: A machine learning model for ranking the perceived affect from motion capture data • m+m: A software framework for building real-time and interactive systems that use movement data *(All contributing to **RQ 2**)*	—
4. Mova: Interactive Movement Analytics Platform	Prototype of an interactive movement analytics framework for feature extraction, feature visualization, and analysis of human movement data (**RQ 2**)	**P 2**
5. AffectNet: Expressive Walking Movement Generation	First iteration of developing an affect-expressive movement generation model (**RQ 3.1**)	**P 3**
6. WalkNet: Interactive Walking Movement Controller	Second iteration of affect-expressive movement generation research, presenting an interactive, affect-expressive walking movement controller (**RQ 3.2** & **RQ 3.3**)	**P 4**
7. GrooveNet 1.0: Music-Driven Dance Generation - Preliminary Results	First iteration of music-driven dance movement generation (**RQ 4**)	**P 5**
8. GrooveNet 2.0: Music-Driven Dance Generation	Second iteration of music-driven dance movement generation (**RQ 4**)	—
9. Summary and Conclusion	Concluding remarks and the future work	—

Table 1.2: Organization of the thesis outlining the appendices and their contributions, as well as their relationship to the research questions. Appendices that represent a published paper are indicated by the corresponding publication number The publication numbers refer to the list presented in Section 1.4.3. "–" indicates the content that has not been previously published.

Appendix	**Contributions**	**Publication**
A. SFU-SIAT Guideline on Cumulative Thesis	SFU-SIAT guidelines regarding a cumulative thesis format	—
B. Supplemental Material: Source Codes	The source code for Mova, AffectNet, WalkNet, GrooveNet 1.0, and GrooveNet 2.0	—
C. Supplemental Material: Outputs from AffectNet, WalkNet, and GrooveNet	Visualizations and animation of outputs from from AffectNet, WalkNet, and GrooveNet	—
D. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space	A machine learning model for ranking the affect from motion capture data (**RQ 2**) -	**P 6**
E. m+m: A novel Middleware for Distributed, Movement based Interactive Multimedia Systems	A software framework for building realtime and interactive systems with movement data (**RQ 2**) -	**P 7**

1.4.2 Thesis Outline

Chapter 1: Introduction
In this chapter, we provide an introduction to this cumulative thesis. We introduce the overarching topic of the thesis and the motivations behind it. We present the research questions, as well as our contributions to them. Next, we outline the structure of the thesis, introduce each chapter's topic, and how it relates to the posed research questions and the rest of the chapters.

Chapter 2: Machine Learning for Data-Driven Movement Generation: a Review of the State of the Art
This chapter provides an extensive review of state of the art on using machine learning techniques and motion capture data to learn and generate movement animation.

Contribution to RQs: This chapter contributes to RQ 1 (see Section 1.3). It outlines the goals and direction of the reviewed literature, lists the available public databases of motion capture data, presents and discuss the advantages and disadvantages of different representations of motion capture data for machine learning applications, discusses the use of a variety of machine learning techniques that are used for movement generation, providing insights on each technique, outlines the techniques for controlling the generation, and lists the gaps and possible directions for the future research in the area.

Chapter 3: Data and Tools for Human Movement Computing
This chapter presents an overview of the data, tools, and libraries that we have developed to support human movement computing research.

Contribution to RQs: This chapter contributes to RQ 2 (see Section 1.3) by introducing the movement data we have collected and published, the web-based front-end that we developed for easy access to mocap data, and the tools such as Mova, PyMO, m+m, and the our work on a model for estimation of affect form motion capture data.

Chapter 4: Mova: Movement Analytics Platform
This chapter presents the prototype of Mova, an interactive, web-based movement analytics platform that allows the user to visualize and analyze multi-modal movement data.

Contribution to RQs: This chapter contributes to RQ 2 (see Section 1.3). We present a movement visualization and analysis platform, using the parallel visual processing capabilities of human perception to visualize multiple movement features in different forms and facilitate a better understanding of the relationships between a particular class of movements and their corresponding measurable features.

Chapter 5: AffectNet: Expressive Walking Movement Generation
This chapter presents the first iteration of creating an affect-expressive movement generation model, including capturing the training data and the design of the machine learning model. The model can control the affective characteristics of the generated movements by manually defining the agent's valence and arousal. Further, we present the result of validating the model's expressive abilities by conducting an experiment where we asked the participants to rate their perceived affective state for both the generated and recorded movements.

Contribution to RQs: This chapter contributes to RQ 3.1 (see Section 1.3) by proposing a model for generating affect-expressive movements controlled by the levels of valence and arousal defined by the user.

Chapter 6: WalkNet: Interactive Walking Movement Controller
This chapter extends the work presented in Chapter 5 in three directions: (1) by incorporating the personal movement signature in the model, (2) by adding navigation capabilities (turning to desired directions) to the model, and (3) by creating an interactive movement controller that allows directing the generated movements using valence and arousal levels, the choice of personal movement signature, and the direction of the walk.

Contribution to RQs: This chapter contributes to RQ 3.2 and RQ 3.3 (see Section 1.3) by proposing an interactive model that allows controlling the personal movement signature and steering the virtual character.

Chapter 7: GrooveNet 1.0: Music-Driven Dance Generation - Preliminary Results
This chapter presents the first iteration of the research on developing a music-driven dance generation model. We trained the model on a relatively small dataset of synchronized music and motion capture recordings of one performer dancing on three songs. The results show that while the model can generate dances for songs that are very similar to the songs that the model is trained on, it fails to generate human-like movements for songs that did not include the training set.

Contribution to RQs: This chapter contributes to RQ 4 (see Section 1.3) by proposing a preliminary model for dance generation using neural networks given a music stream.

Chapter 8: GrooveNet 2.0: Music-Driven Dance Generation
This chapter builds on the work presented in Chapter 7 by (1) extending the training dataset with recordings from nine more performers, and (2) by proposing a more robust machine learning model that can generate dance movements for songs that are not in the training set. We analyze and discuss the performance and characteristics of the model under various input conditions.

Contribution to RQs: This chapter contributes to RQ 4 (see Section 1.3) by extending the training dataset, proposing a new model for music-driven dance generation, discussing the extent of the model's capabilities, and outlining our plans on improving the model and using the system for an interactive art installation.

Chapter 9: Summary and Conclusion
This chapter puts together a discussion of the presented works and outlines future work directions on this topic.

Appendix A: SIAT Guidelines on Cumulative Thesis
This appendix presents the SFU-SIAT guidelines regarding a cumulative thesis format.

Appendix B: Supplemental Material: Source Codes
This appendix documents the source code for Mova (Chapter 4), AffectNet (Chapter 5), WalkNet (Chapter 6), and both GrooveNet iterations (Chapters 7 and 8).

Appendix C: Supplemental Material: Outputs from AffectNet, WalkNet, and GrooveNet
This appendix presents animation and visualizations of the outputs from each of the proposed models.

Appendix D: Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space
This appendix presents the publication on affect estimation of motion capture data. It is related to the research on affect-expressive movement generation presented in Chapter 5.

Appendix E: m+m: A novel Middleware for Distributed, Movement-based Interactive Multimedia Systems
This appendix presents the publication on the m+m: Movement + Meaning middleware project. m+m is an opensource framework for building real-time pipelines for movement data.

1.4.3 Publications

P1 *Omid Alemi and Philippe Pasquier. 2019. Machine Learning for Data-Driven Movement Generation: a Review of the State of the Art. CoRR abs/1903.08356 (2019). arXiv:1903.08356* `http://arxiv.org/abs/1903.08356`

Note about the collaboration: Omid Alemi collected the related work and wrote the publication. Philippe Pasquier was responsible for research direction, research design, and project management.

P2 *Omid Alemi, Philippe Pasquier, and Chris Shaw. 2014. Mova: Interactive Movement Analytics Platform. In Proceedings of the 2014 International Workshop on Movement and Computing (MOCO'14). ACM, 37–42.*

Note about the collaboration: This research was done as the final project for the Data Visualization course, taught by Chris Shaw in the Fall of 2013. Omid Alemi worked on the ideation, development, and writing of the publication with the guidance of both Chris Shaw and Philippe Pasquier.

P3 *Omid Alemi, William Li, and Philippe Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII). 442–448.*

Note about the collaboration: Omid Alemi worked on the data collection, the machine learning research, and writing the majority of the publication. William Li contributed by conducting the validation study and writing the relevant content in the publication. Philippe Pasquier was responsible for research direction, research design, and project management.

P4 *Omid Alemi and Philippe Pasquier. 2017. WalkNet: A Neural-Network-Based Interactive Walking Controller. In Proceedings of the 17th International Conference on Intelligent Virtual Agents (Lecture Notes in Computer Science), Vol. 10498. Springer, 15–24.*

Note about the collaboration: Omid Alemi designed and developed the project and wrote the publication. Philippe Pasquier was responsible for research direction, research design, and project management.

P5 *Omid Alemi, Jules Françoise, and Philippe Pasquier. 2017. GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks. Poster accepted to the Work- shop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining - https://ml4creativity.mybluemix.net/.*

Note about the collaboration: Omid Alemi developed and conducted the machine learning research and wrote the majority of the publication. Jules Françoise contributed by working on the music feature extraction algorithms and writing the publication's

related content. Philippe Pasquier was responsible for research direction, research design, and project management.

P6 *William Li, Omid Alemi, Jianyu Fan, and Philippe Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In Proceedings of the 5th International Conference on Movement and Computing (MOCO '18). ACM, Article 18, 8 pages.*

Note about the collaboration: William Li wrote the majority of the publication and designed and ran the ground-truthing study of the data in collaboration with Jianyu Fan. Omid Alemi contributed by training and testing the machine learning model. Philippe Pasquier was responsible for research direction, research design, and project management.

P7 *Ulysses Bernardet, Dhruv Adhia, Norman Jaffe, Johnty Wang, Michael Nixon, Omid Alemi, Jordon Phillips, Steve DiPaola, Philippe Pasquier, and Thecla Schiphorst. 2016. M+m: A Novel Middleware for Distributed, Movement Based Interactive Multimedia Systems. In Proceedings of the 3rd International Symposium on Movement and Computing (MOCO '16). ACM, Article 21, 9 pages.*

Note about the collaboration: This was a collaborative project with Thecla Schiphorst as the Principal Investigator and Ulysses Bernardet as the lead post-doctoral researcher, along with Dhruv Adhia and Norman Jaffe as industry collaborators, Omid Alemi as researcher and developer, Johnty Wang, Michael Nixon, and Jordon Phillips as developers, and Steve DiPaola and Philippe Pasquier as supervisors. Omid Alemi contributed by integrating MoDa and Mova into the framework.

1.5 Summary

In this chapter, we first introduced the topics of movement perception, movement manipulation, and movement generation within the field of human movement computing, a growing field of research with no agreed-upon foundation. After explaining the three topics and pointing out their challenges and use-cases, we noted that in this thesis we focus on movement generation and explained the motivations behind our research. Next, we listed our research questions and our contributions to them. Finally, we provided an outline of this thesis as a guide on how to read each chapter, clarified the contributions of collaborators, and listed the publications that are presented in this thesis.

Bibliography

O. Alemi, P. Pasquier, and C. Shaw. 2014. Mova: Interactive Movement Analytics Platform. In *Proceedings of the 2014 International Workshop on Movement and Computing (MOCO '14)*. ACM, 37–42.

U. Bernardet, D. Adhia, N. Jaffe, J. Wang, M. Nixon, O. Alemi, J. Phillips, S. DiPaola, P. Pasquier, and T. Schiphorst. 2016. m+m: A Novel Middleware for Distributed, Movement Based Interactive Multimedia Systems. In *Proceedings of the 3rd International Symposium on Movement and Computing (MOCO '16)*. ACM, Article 21, 9 pages.

Y. Candau, J. Françoise, S. F. Alaoui, and T. Schiphorst. 2017. Cultivating Kinaesthetic Awareness through Interaction: Perspectives from Somatic Practices and Embodied Cognition. In *Proceedings of the 4th International Conference on Movement Computing (MOCO '17)*. ACM, Article 21, 8 pages.

T. Clarke, M. Bradshaw, D. Field, S. Hampson, and D. Rose. 2005. The Perception of Emotion from Body Movement in Point-Light Displays of Interpersonal Dialogue. *Perception* 34, 10 (2005), 1171–1180. PMID: 16309112.

N. Fourati and C. Pelachaud. 2018. Perception of Emotions and Body Movement in the Emilya Database. *IEEE Transactions on Affective Computing* 9, 1 (2018), 90–101.

T. Geijtenbeek and N. Pronost. 2012. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. *Computer Graphics Forum* 31, 8 (2012), 2492–2515.

Google ATAP. 2020. Project Soli. Retrieved 2020-10-30 from `https://atap.google.com/soli/`

M. Karg, A.-A. Samadani, R. Gorbet, K. Kuhnlenz, J. Hoey, and D. Kulic. 2013. Body Movements for Affective Expression: A Survey of Automatic Recognition and Generation. *IEEE Transactions on Affective Computing* 4, 4 (2013), 341–359.

D. Kirsh. 2013. Embodied Cognition and the Magical Future of Interaction Design. *ACM Transactions on Computer-Human Interaction* 20, 1, Article 3 (April 2013), 30 pages.

W. Li, O. Alemi, J. Fan, and P. Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In *Proceedings of the 5th International Conference on Movement and Computing (MOCO '18)*. ACM, Article 18.

W. Li and P. Pasquier. 2016. Automatic Affect Classification of Human Motion Capture Sequences in the Valence-Arousal Model. In *Proceedings of the 3rd International Symposium on Movement and Computing*. ACM, Article 15.

T. Pejsa and I. s. Pandzic. 2010. State of the Art in Example-Based Motion Synthesis for Virtual Characters in Interactive Applications. *Computer Graphics Forum* 29, 1 (2010), 202–226.

Teenage Engineering. 2020. R. Retrieved 2020-10-30 from https://teenage.engineering/designs/R

N. F. Troje. 2002. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision* 2, 5 (09 2002), 2–2.

N. F. Troje. 2008. *Retrieving information from human movement patterns.* Oxford University Press, Chapter 12, 308–334.

S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges. 2016. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16).* ACM, 851–860.

X. Wang, Q. Chen, and W. Wang. 2014. 3D Human Motion Editing and Synthesis: A Survey. *Computational and Mathematical Methods in Medicine* 2014, 3 (June 2014), 1–11.

A. Wilson and S. Golonka. 2013. Embodied Cognition is Not What you Think it is. *Frontiers in Psychology* 4 (2013), 58.

Chapter 2

Machine Learning for Data-Driven Movement Generation: a Review of the State of the Art

Abstract

In this survey, we review and analyze different aspects of building automatic movement generation systems using machine learning techniques and motion capture data. We cover topics such as high-level movement characterization, training data, features representation, machine learning models, and evaluation methods. We conclude by presenting a discussion of the reviewed literature and outlining the research gaps and remaining challenges for future work.

2.1 Introduction

One of the most common ways to produce realistic humanoid movement animation is through motion capture of human performers. Along with the proliferation of interactive storytelling mediums such as web, virtual and augmented reality, the shift from linear media (e.g., music, books, movies, etc.) to non-linear media (e.g., video games, interactive installations, etc.) has resulted in an increase in the need for creating diverse content, ranging from sound and music to humanoid movement animation.

In particular, the dynamic and interactive nature of non-linear applications lead to a need for the animation of anthropomorphic virtual agents with a wide range of behaviours, actions, expressions, and personalities. This increase in demand is changing the practice of creating movement animation as the traditional methods are too costly and time consuming to be used in non-linear applications (Tomlinson, 2005; Pejsa and Pandzic, 2010). As a result, a body of research around building automatic movement generation models is built over the past two decades.

Motion capture data are used in a variety of data-driven movement animation generation techniques, ranging from sequence concatenation (e.g., (Tanco and Hilton, 2000)) and sequence blending (e.g., (Kwon and Shin, 2005; Hsu et al., 2005)) to machine learning models. In this paper, we focus on the data-driven approaches that use motion capture data to train machine learning models that can generate movement animation of humanoid characters. To limit the scope of the paper, we do not include physics-based techniques that also incorporate machine learning and motion capture data. Furthermore, we focus on animated software agents, although we note that the techniques used for both types of agents are not mutually exclusive and similar models can be used in the motor controllers of physical robots and software agents, e.g., (Herzog et al., 2008; Matsubara et al., 2010; Kulić et al., 2011) (13) (15) (19)[1].

Machine learning models are not confined to the variations that exist in the training data and can be used to learn a generalized space of movements, fill the missing data, or generate continuous streams of movements. The potential for the application of using machine learning models in movement animation generation can be demonstrated by the success of such models in generating patterns of data in other fields, such as speech synthesis, e.g., (Zen et al., 2012; Ling et al., 2013), computer music, e.g., (Briot et al., 2019), and visual textures, e.g., (Kivinen and Williams, 2012).

Review studies have covered the physics-based and data-driven techniques for movement generation in general. For example, Wang et al. (2014) present a general overview of the field of 3D human movement editing and generation. Geijtenbeek and Pronost (2012) provide an extensive survey of different components of physics-based models and review the literature.

[1]Numbers in (num) refer to the items in the tables.

In another study, Pejsa and Pandzic (2010) review the literature on data-driven methods for creating graph-like structures, motion planning, and parametric movement synthesis using interpolation techniques, but do not specificly cover machine learning models. Karg et al. (2013) review the recognition and generation techniques in the domain of affect-expressive movements. While building generative models using machine learning and motion capture data has been actively researched over the past two decades, to the best of our knowledge, there is no study presenting a comprehensive survey of the literature on applying machine learning models on motion capture data for animation generation. The scope of this paper differs from the aforementioned reviews as it focuses on the implications of processing motion capture data, challenges in training generative machine learning models, and the techniques on controlling the movement generation.

In summary, the contributions of this paper are as follows:

- We identify the key goals, challenges, and gaps in the research on machine-learning-based movement generation.
- We present a framework for the characterization of movement in the literature.
- We provide an elaborate discussion on the use of different machine learning techniques with motion capture data for learning and generating movement animation.

The rest of the paper is organized as follows: we first review the typical architecture of a generative system such as those reviewed here, and outline the goals, challenges, and the design choices that are involved, in Section 2.2. We summarize our findings on the characterization of movement in the literature and detail our framework in Section 2.3. We discuss recording, processing, and representations of movement data in the literature, and list the publicly available movement databases in Section 2.4. We survey the application of using machine learning techniques with motion capture data for learning and generating movement animation in Section 2.5. We look at the evaluation methods in Section 2.6. We summarize our findings and provide a discussion of the gaps and remaining challenges in Section 2.7. Finally, we present our conclusions in Section 2.8.

2.2 Background and Fundamentals

In this section we present the fundamentals of machine-learning-based movement generation. We first lay out the definitions and assumptions that we use throughout this paper. Next, we describe a typical architecture for capturing, learning, and generating human movement. We discuss the common themes and the research goals motivating the field, followed by a description of the applications of movement generation.

2.2.1 Definition of Key Concepts

- **Skeleton:** In modelling and animating full-body movement, it is common to use a skeleton to represent the body structure. Each body pose is described by a set of the rotations of the joints (or equivalently the bones), as well as the orientation and position of the agent in the global space (typically called the *root*). The hierarchy of the joints and their rotations are constrained by a pre-defined skeleton structure.

- **Posture / Pose:** Posture or pose refer to a static state of the body, described by the positions or orientations of the body parts as a whole. Numerically, a pose is represented by a single frame of data.

- **Pose Space:** We use the term pose space to refer to the space of all possible body poses. A pose can then be interpreted as a single point in this space.

- **Gesture:** Gesture is the movement of a subset of body parts, often performed to communicate information (Lamb, 1965).

- **Movement:** By movement we refer to the animation of a full-body representation of an anthropomorphic skeleton through time.

- **Motion:** We make a distinction between human *motion* and human *movement*. While mostly used interchangeably in the literature, we use *motion* to refer to the changes of the position of a single entity in the space (e.g., the body as a whole or individual body limbs). In contrast, we use human *movement* to refer to the coordinated motion of individual body limbs.

- **Movement Primitives:** The notion of *movement primitive* is used to represent basic segments of human movements that constitute longer movements (Schaal et al., 2003).

- **Factor Space:** We use the term factor space to refer to a high-level space of movement descriptors, such as those describing actions and emotions.

- **Agent:** We use the term agent to refer to an abstract model of a mover. Although the agent can refer to a human, physical robot, or a software, throughout the paper we use agent to specifically describe software agents.

- **Mover:** Throughout this paper, we use the terms *actor*, *mover*, *dancer*, *performer*, and *subject*, interchangeably to refer to the person or agent moving.

- **Personal Movement Signature/Style:** An individual's distinguishable movement patterns that is influenced by a combination of factors such as the individual's physical build and cultural background (Bartenieff and Lewis, 1980; Serlin et al., 2007).

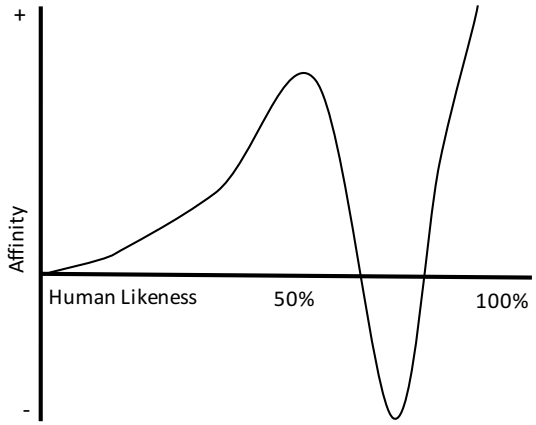


Figure 2.1: The uncanny valley: the relationship between people's affinity towards human-like agents, as they approach human likeness. Reproduced based on (Mori et al., 2012).

2.2.2 Research Directions

We categorize the directions that the body of research on movement generation follows in three themes: (a) achieving believability, (b) controlling and manipulating the characteristics of the generated movements, and (c) supporting real-time and continuous generation. Each of the themes bring challenges that justify the design and development of movement generation models.

(a) *Believability*: Believability is one of the fundamental notions in virtual agent animation (Lasseter and Lasseter, 1987; Bates, 1994; Pejsa and Pandzic, 2010; Mori et al., 2012). Even non-movement-expert humans notice the smallest details that make movement look unnatural (Pejsa and Pandzic, 2010; Geijtenbeek and Pronost, 2012). It is challenging to manually create a believable animation that looks appealing to the audience from scratch.

The animation industry has employed motion capture technologies in order to record the movements of human actors. The recordings preserve the realism and expressive details of the movement, and is used as the basis of animations (Menache, 2000). Data-driven movement generation methods also take advantage of motion capture data in order to create natural-looking animations (Pina et al., 2000). The challenge facing such methods is the often unwanted noise or artifacts that are introduced as a result of the computational manipulations of the recorded data. Generating natural-looking movements is therefore one of the intrinsic goals of data-driven movement generation techniques.

There are two ingredients that are essential in achieving higher levels of believability in movements of an agent:

– *Physical Validity*: As in reality, humans move in a physical world, their movements are constrained by the laws of physics. The movements that are generated also need to follow the laws of physics and the biomechanics that are involved in human movement. Note that in humans, the notion of physics is implicit. The brain does not explicitly solve physical equations in order to produce the movement patterns. However, trough feedback loops from the physical environment, the brain learns to adapt to the laws of physics.

  As most of the data-driven methods approach the modelling problem without any prior assumptions about the mechanisms that produce the data, they are limited in guaranteeing obeying the laws of physics. Hybrid approaches, e.g., (Wei et al., 2011; Liu and Hodgins, 2018), combine data-driven methods with physical models to generate movements that are physically valid and look natural.

  The majority of the data-driven studies do not address the problem of physical validity. Although by learning the movements from real data and imitating their qualities, the generated movements uphold some of the physical properties of human movements, there is no constraint to enforce such rules and react to dynamic changes of forces.

– *Expressivity*: The expressiveness of the movements of agents plays an important role in their believability (Bates, 1994). Movement is a form of non-verbal communication and conveys affective qualities that reflect the inner state of the agent (Troje, 2002, 2008). A generative model of movement, should therefore be able to exhibit a variety of expressions, and allow controlling those expressions according to some high-level descriptors. In particular, the literature on data-driven models has addressed modelling expressive walks (Tilmanne and Dutoit, 2010; Tilmanne et al., 2014; Alemi et al., 2015) (16) (27) (29) and hand movements (Taubert et al., 2011, 2012; Samadani et al., 2013) (22) (22) (26). Modelling the expressivity of movement is discussed in more details in Section 2.3. Expressivity is one of the advantages that data-driven models have over physics-based movement modelling approaches.

Note that achieving a high level of realism may not always results in the agent being perceived as natural, which is a concept known as the *uncanny valley* introduced by Mori et al. (2012). As shown in Figure 2.1, people's affinity to human-like animated agents or robots increases as the similarities to real humans increases, but it abruptly diminishes as the similarities reach to near human-like but fail to reach too close. With respect to data-driven movement generation techniques, one could argue that achieving the same level of realism as the original recorded data is a satisfying criteria for evaluating the naturalness of the movement. We discuss the evaluation methods of movement generation systems in Sections 2.6 and 2.7.

(b) *Control and Manipulation*: Automatic movement generation is fully utilized when provided with a level of control over the characteristics of the movements being generated (Chai et al., 2002; Pejsa and Pandzic, 2010; Geijtenbeek and Pronost, 2012). The ability to control and manipulate the movement is one of the main elements that give data-driven, and statistical generation techniques in particular, benefit over using just the recorded movements. A single model can generate many variations of the same movement, while one would otherwise need to capture all those variations individually, and blend or sequence them manually.

An agent's movements portray its personality, emotional state, goals, and intentions, while corresponding to its reactions to the external stimuli from the environment and other agents surrounding it (Bartenieff and Lewis, 1980; Studd and Cox, 2013). One can therefore use high-level cognitive attributes and states in order to control and manipulate the generated movements.

These many sources of influence result in a large combinatorial space of possible movements. Consequently, the problem of manipulation and control is nontrivial and it brings challenges and requirements that we detail below and throughout this paper.

- *Movement Parameterization*: Directly manipulating movement at the level of raw data (joint rotations) is cumbersome and inefficient, mostly due to the low-level, high-dimensional, dense in time, and non-linear space of movement data. It is easier to manipulate a high-level representation that is sparse in time and has fewer dimensions than the raw data. In addition, it is easier to associate a high-level representation with the meaningful characteristics of the agent and its movement. This has motivated the research on learning a mapping from a low-dimensional control space to the high-dimensional pose space, as well as performing operations such as interpolation and extrapolations on the parameters. Techniques for addressing these are described in details in Sections 2.5 and 2.7.
- *Characterization of Movement*: In order to properly integrate the movement generation process into an agent with physiological and psychological properties, the high-level parameters used to control the movement have to correspond to, directly or indirectly, the agent's physiological, mental, emotional states, components, as well as the properties of the environment in which the agent resides . As will be discussed in further details in Sections 2.3 and 2.4, most studies have not adopted a characterization framework that refers to meaningful concepts and either model a single pattern of movement, or model arbitrary variations of a movement pattern.
- *Motor Variability:* Various studies have shown that variability is a fundamental characteristic of the movement of biological entities including humans (Davids et al., 2006; Müller and Sternad, 2009). Humans never exactly repeat the same

movement even when they try to do so. In other words, although multiple repetitions of the same movement can have the exact same functional, planning, and expressive descriptors, the execution dimension will always differ for each try. Thus, models that replicate the same execution will be perceived as more mechanical than natural. For example, Motion Graphs (Kovar et al., 2002) and similar approaches use exact copies of the recorded motion capture segments (except some of the transitions), which replicate the same execution over and over.

(c) *Interactive and Real-Time Animation:* Generating movement animation interactively, as in applications such as video games, requires two conditions to be present:

- *Computational Constraints:* A model that generates new samples in real-time given a set of parameters makes it possible to be used in interactive applications, in which real-time generation of the contents is desired. Real-time generation brings challenges in both time and space complexities of the generation algorithms of a statistical model. The model should be able to generate new frames according to the frame-rate of the system, while leaving enough processing power and memory for other computations needed in the system.
- *Generate Transitions*: Interactive animation requires making a large number of transitions between consecutive movements segments. Due to the dynamic nature of the scenarios, the exact timing and occurrences of such transitions cannot be defined and authored by the animator a priori. Therefore, the transitions need to be generated in real-time. While generating a transition can be seen as simply blending the source and target movements, a statistical model that has learned a general model of movement can be able to generate transitions the same way it generates any movement. Creating smooth and believable transitions is more challenging than generating movement segments with fixed characteristics and remains an open problem, which is discussed in Sections 2.3 and 2.5.

As described above, automatic movement generation research follows the goals along three general themes of believability, control and manipulation, and interactive animation. In the next part, we argue that the aforementioned research goals are shaped by two types of applications for automatic movement generation.

2.2.3 Learning and Generating Movement

A typical machine-learning-based generative system synthesizes new movements by learning a movement model from a group of recorded movement segments. The generic architecture of such a system is shown in Figure 2.2. In this section, we briefly highlight different parts of the architecture, while leaving their detailed discussions in the proceeding sections.

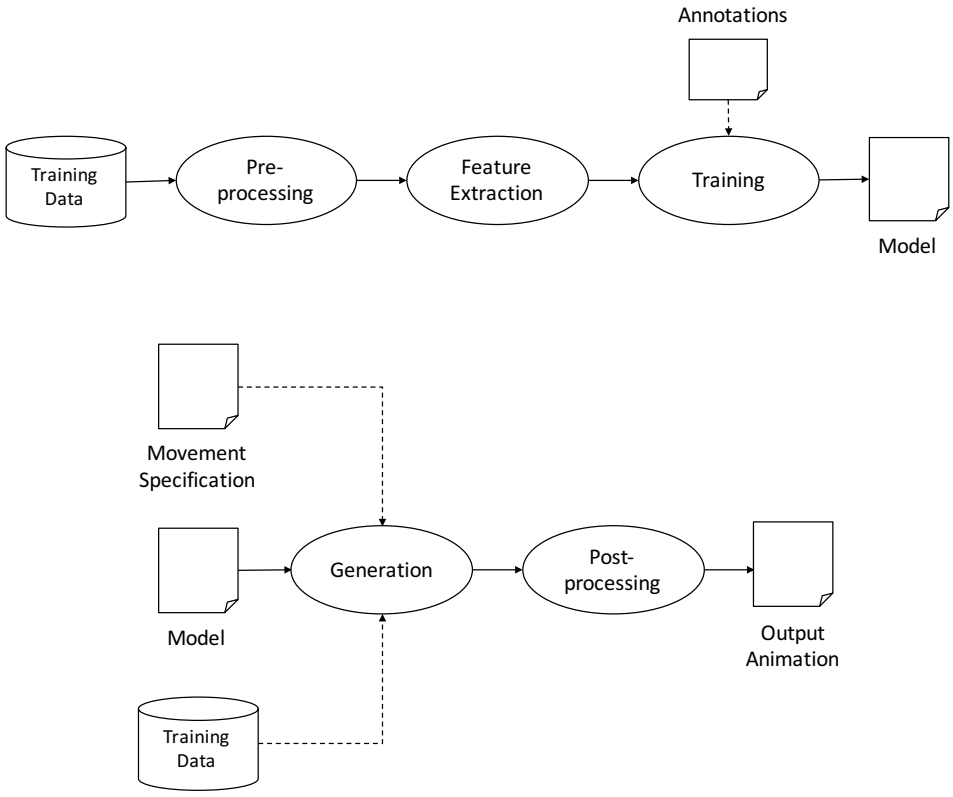


Figure 2.2: Training the model (top), and generating new movements (bottom).

The first choice in designing a generative model is the type of the movements to generate. In data-driven approaches, the repertoire of the movements that a model can generate relies on the diversity of the movements that exist in the training data, the available variations of each movement type, the number of samples for each variation, and the number of human actors performing these samples. The size of the training dataset is important. If trained on a relatively small dataset, a machine learning model might closely imitate the only movements it has seen, but fail to learn a more general space of possible movements. In case of supervised learning, it is required to annotate the training data based on some descriptors (Section 2.4.2).

Depending on the choice of the movements, one can use the data available from a public database (Section 2.4.3), or record the movements using a motion capture system (Section 2.4.1). The latter is costly and time-consuming, but could produce a more desirable set of data, while the former data are ready to use, but might not directly fit the desired requirements of a particular study.

In the optional pre-processing and feature extraction stage, the raw training data is transformed to a set of features to make them more suitable to be used by the learning algorithm (Section 2.4.1). The pre-processing stage can include changing the rotational

representation of the data; calculating the joint speed and acceleration, or learning more suitable representations of the raw input data using feature learning or extraction algorithms; or reducing the dimensionality of the data. The training sequences might also be divided into shorter segments.

If multiple databases are combined, each data source might use a different skeleton with a different shape, size, and number of body joints. In most cases, one needs to re-target the data to a uniform representation of the data from all sources so that they can be interpreted by the machine learning model in the same way.

In the training stage, depending on the machine learning technique and the type of the features used, a learning algorithm is employed to determine the generation function. In some cases, more than one learning algorithm might be used for different parts of the system. The learning can be supervised, unsupervised, or semi-supervised. In supervised learning, the training process involves learning the correlation between the movement data and a set of labels. In unsupervised learning, the data is not labelled and the model learns the underlying patterns that generate the data. In semi-supervised learning, only a subset of the training data is labelled and the training process involves both supervised and unsupervised techniques. The choice of which learning method to use is determined by the problem being addressed, as well as the approach designed to tackle the problem. We will discuss the examples and implications of each method throughout the paper.

A generation algorithm (Section 2.5) uses the learned model to create new samples. A group of models are able to generate new data based on a given description of the movements, which allow controlling the qualities of the generated movements (Section 2.2.2). Some machine learning techniques such as Gaussian Process Models need to retain the training data to be able to generate new samples while others require to keep only a few initialization frames (such as some artificial neural networks) or do not need any data for the generation (such as Hidden Markov Models).

The raw output of the model goes through the post-processing stage to be converted to a movement representation that can be used for animation. It is often the case that the post-processing involves reversing the steps performed in the pre-processing stage.

The quality of the output of the system is then evaluated formally or informally, as discussed in (Section 2.6).

2.3 Characterization of Movement

Movement is multifaceted. Multiple elements influence the movement: the internal state of the agent performing the movement (e.g., emotions and intentions), as well as the external stimuli that shape the environment surrounding the agent (e.g., objects, gravity, friction, etc.).

We use the term "factors" to describe *the sources of influence* on the agent movement. Each factor has a specific domain, which can be continuous or discrete. Choosing different values for a factor results in movements with different characteristics. For example, if we consider the position of the hand as a factor, the factor space would be the 3D space that is within the reach of the hand. Or if we consider the affective state of the agent as a factor and follow a categorical representation of affect, the categories such as happy, sad, or afraid would be within the space of the factor.

The interaction and combination of the factors across multiple dimensions (e.g., affective state, actions, etc.) result in the endless varieties of movement that humans can perform. As building movement generation systems that can understand and generate all of this endless variety is not yet feasible (see the discussion in Section 2.7), researchers choose a subset of movements to model, and only a few factors to describe these movements (if any.)

In this section, we first review and criticize the characterization of movement and its factors in the literature. Next, we present a framework to characterize movement based on the factors that are meaningful to agents.

2.3.1 Movement Factors in the Literature

We present a summary of the dimensions of movement that are characterized, the definitions and application of the factors, and the controlling abilities of the reviewed works in Table 2.1.

The majority of the works address movement characterization from a perceptual perspective, i.e., how an arbitrary factor changes the perceived movement, rather than from an agency perspective, i.e, how factors that characterize an agent's internal state change the movement.

Research on perceptual systems has identified the notion of *content* and *style* as two factors of a perceptual system (Tenenbaum and Freeman, 2000). For instance, the same word (the content) can be spoken in different accents (the style), or the same letter (the content) can be written with different fonts (the style). Although movement is not merely a perceptual system, style and content separation is applied to the domain of human movement analysis. For instance, *walking* from point A to point B in an environment, the *content*, can be performed in different *styles*, such as taking different paths, exhibiting distinct movement signatures, or expressing different emotions.

Consequently, the research on statistical movement generation has adopted the concept of *style and content separation* as a method for controlling the characteristics of the generated movements, .e.g., (Wang et al., 2006a; Taylor et al., 2006; Herzog and Krüger, 2009; Tilmanne et al., 2014; Alemi et al., 2015) ⑥ ⑧ ⑬ ㉗ ㉙, creating new styles for movements, e.g., (Chiu and Marsella, 2011a; Tilmanne et al., 2014) ㉑ ㉗, or transferring the style of one movement to another, e.g.,(Brand and Hertzmann, 2000; Wang et al., 2007) ① ⑨.

Table 2.1: An overview of the characterization of movement in the reviewed literature.

	Ref	ML Technique	Characterization	Control Support	Control Technique	Discrete / Continuous Factors	Supervised / Unsupervised Learning	Single / Multi Factor(s)	Modelled Movements
①	Brand and Hertzmann (2000)	Stylistic HMM	Expression - Personal Signature	Yes	Parametric Gaussian Distribution	C	U	M	Gender, weight distribution, grace, energy, and formal dance styles
②	Tanco and Hilton (2000)	HMM	Function	Yes	Hierarchical architecture and clustering	D	U	S	Standing up, walking, running
③	(Li et al., 2002)	LDS	Expression	No	-	-	-	-	Disco dance
④	Yamazaki et al. (2005)	HSMM + Multiple Regression	Expression	Yes	Parametric Gaussian Distribution	C	S	M	Walking with different speed and stride length
⑤	Wang et al. (2005)	HMM	Function	Yes	Hierarchical architecture	D	U	S	Regular walk, chopping a tree, ballet walk, ballet roll, disco, and complex disco
⑥	Wang et al. (2006a)	SOMN	Planning	Yes	Parametric Gaussian Distribution	C	S	M	Boxing, with varying body height and the distance of punch target
⑦	(Wang et al., 2006b)	HMM/Mix-SDTG	Planning	Yes	Parametric Gaussian Distribution	C	S	M	The height of the right arm
⑧	Taylor et al. (2006)	CRBM	-	No	-	-	-	-	Walking and running
⑨	(Wang et al., 2007)	Multifactor GPLVM	Expression	Yes	Mapping points in a low-dimensional latent space to movement	C	Semi S	M	Identity and gait in walking
⑩	(Wang et al., 2008)	GPDM	-	No	-	-	-	-	Walking
⑪	(Lin et al., 2008)	MLP	Planning	Yes	Regression	C	S	M	Humanoid arm movements
⑫	Qu et al. (2008)	Isomap + LDS	Function and Expression	No	-	-	-	-	Boxing, Indian dance
⑬	(Herzog et al., 2008; Herzog and Krüger, 2009)	Parametric HMM	Planning	Yes	Interpolating individual models for each parameter value and using parametric Gaussian distribution	C	S	M	the position of the pointing target
⑭	Taylor and Hinton (2009)	Factored CRBM	Expression and Planning	Yes	Modulating network weights	C	S	M	Walking styles + walking speed and stride length
⑮	Matsubara et al. (2010)	Nonlinear Dynamical Systems	Planning	Yes	Dynamical systems	C	S	M	Table tennis
⑯	Tilmanne and Dutoit (2010)	PCA	Expression	Not explicit	Principal Components	-	-	-	Walking styles
⑰	(Wei et al., 2011)	GP + Physics	-	No	-	-	-	-	Running, walking, and jumping under different physical forces

Table 2.1: An overview of the characterization of movement in the reviewed literature - *Continued*.

	Ref	ML Technique	Characterization	Control Support	Control Technique	Discrete / Continuous Factors	Supervised / Unsupervised Learaning	Single / Multi Factor(s)	Modelled Movements
(18)	(Liu et al., 2011)	Multilinear ICA	Planning and Expression	Yes	Optimization	C	S	M	Sideways stepping, reaching, and striding over obstructions, with multiple actors
(19)	(Kulić et al., 2011)	HMM	Function	Yes	Hierarchical model	-	-	-	Arm raising, bending, walking, squating, kicking
(20)	Chiu and Marsella (2011b)	Hierarchical FCRBM	Execution	Yes	Modulating network weights	C	S	M	Prosody of speech in gestures
(21)	Chiu and Marsella (2011a)	Hierarchical FCRBM	Expression	Yes	Modulating network weights + blending networks	D	S	S	Walking styles
(22)	Taubert et al. (2011, 2012)	Hierarchical GPLVM + HMM	Expression	Yes	Model interpolation	D	S	S	Handshake
(23)	Tilmanne et al. (2012)	HMM + Transformation	Expression	No	Model interpolation + Transformation algorithms	-	S	S	Walking styles
(24)	Min and Chai (2012)	Functional PCA + Gaussian Mixture Model + Gaussian Process	Function and Planning	Yes	Graphs + Optimization	C	S	M	Function transitions: walking, sitting, picking, placing; Controling the movement: direction, end-position, speed
(25)	Levine et al. (2012)	GPLVM	Planning	Yes	Markov Decision Processes	C	S	M	Locomotion and karate
(26)	Samadani et al. (2013)	Functional PCA	Expression	Not explicit	Principal components + clustering	-	U	-	Hand movement
(27)	Tilmanne et al. (2014)	HSMM	Expression	Yes	Model interpolation	D	S	S	Walking with different emotions, morphology personifications, or situations
(28)	Fragkiadaki et al. (2015)	Recurrent Neural Networks	Function	No	-	-	-	-	Waling, eating, smoking
(29)	Alemi et al. (2015)	Factored CRBM	Expression	Yes	Modulating network weights	C	S	M	Valence and arousal dimensions of affect
(30)	Jain et al. (2015)	Recurrent Neural Networks	-	No	-	-	-	-	Walking, eating, smoking, and having a discussion
(31)	(Crnkovic-Friis and Crnkovic-Friis, 2016)	LSTM RNN	Expression	No	-	-	-	-	Contemporary Dance
(32)	(Holden et al., 2016)	Convolutional Autoencoders + Feed-forward Networks	Planning	Yes	Using a *control* neural network	C	Semi S	M	Navigation, punching and kicking, factor transfer, crowd animation

Table 2.1: An overview of characterization of movement in the reviewed literature - *Continued*.

	Ref	ML Technique	Characterization	Control Support	Control Technique	Discrete / Continuous Factors	Supervised / Unsupervised Learaning	Single / Multi Factor(s)	Modelled Movements
(33)	Wang and Artières (2017)	Adverserial Learning	Function and Expression	Yes	Factors as conditional inputs	C	S	M	Experiments
(34)	Herrmann et al. (2017)	Functional PCA + Gaussian Mixture Model + Gaussian Process + kMeans Trees	Function, Planning, and Expression	Yes	Graphs + Optimization	C	S	M	Walking with different emotions, lifting, sitting, moving books, knocking on the door, throwing
(35)	Martinez et al. (2017)	Recurrent Neural Networks	Function	No	-	-	-	-	Mean angle error for prediction of a variety of actions
(36)	Alemi and Pasquier (2017)	Factored FCRBM	Planning and Expression	Yes	Modulating network weights	C	S	M	Valence and arousal dimensions of affect, walking direction, movement signature
(37)	Alemi et al. (2017)	Factored FCRBM	Expression	Yes	Modulating network weights	C	S	M	Dance movements for a given song
(38)	Li et al. (2017)	Recurrent Neural Networks	-	No	-	-	-	-	Martial arts, Indian dance, Indian/salsa hybrid dance, walking
(39)	Holden et al. (2017)	Fully-Connected Networks with dynamically generated weights	Planning	Yes	Generating the weights of the network based on the control parameters	C	S	M	Locomotion
(40)	Lee et al. (2018)	Recurrent Neural Networks	Function and Planning	Yes	Extra input + specialized loss functions	C	S	M	Basketball, tennis, locomotion including walking, turning, side stepping, and backward stepping
(41)	Pavllo et al. (2018)	Recurrent Neural Networks	Planning	Yes	Training an extra control network	C	S	M	Short-term generation with no control: various movements - Long-term generation with control: locomotion
(42)	Li et al. (2018)	Convolutional Networks and Autoencoders	-	No	-	-	-	-	Walking, eating, smoking, and having a discussion
(43)	Starke et al. (2019)	Fully-Connected Networks with dynamically generated weights	Planning	Yes	Generating the weights of the network based on the control parameters	C	S	M	Character's interactions with the environment (walking, running, sitting, opening doors, picking up, carrying, and putting down boxes)
(44)	Wang et al. (2019)	Recurrent Neural Networks	Planning	Yes	Maximum A Posteriori (MAP) and optimization	C	S	M	Walking and running with different speeds, step lengths, and turning angles

Table 2.1: An overview of characterization of movement in the reviewed literature - *Continued*.

	Ref	ML Technique	Characterization	Control Support	Control Technique	Discrete / Continuous Factors	Supervised / Unsupervised Learaning	Single / Multi Factor(s)	Modelled Movements
(45)	Pettee et al. (2019)	Autoencoders	Expression	Indirect Control	Latent space	-	U	-	Improvisational dance
(46)	Sun et al. (2021)	Recurrent Neural Networks with Generative Adversarial Networks	Expression	Yes	Extra Input	C	S	M	Dancing to music
(47)	Starke et al. (2020)	Fully-Connected Networks with dynamically generated weights	Planning	Yes	Generating the weights of the network based on the control parameters	C	S	M	Basketball
(48)	Ling et al. (2020)	Fully-Connected Autoencoders	Expression	Yes	Extra Input	C	S	M	Locomotion

Employing the two-dimensional style and content characterization of movement brings out the question of what is considered the content of a movement and what is considered its style. A majority of studies have considered gaits variations during locomotion, or anthropomorphization of non-human creatures as the *style* factor of movement, treating the locomotion as the *content* factor, e.g., (Taylor and Hinton, 2009; Tilmanne and Dutoit, 2010; Tilmanne et al., 2012; Chiu and Marsella, 2011a; Tilmanne et al., 2014) (14) (16) (23) (21) (27). Other studies consider gender or the personal movement signature as the stylistic factors, e.g, (Brand and Hertzmann, 2000; Wang et al., 2007) (1) (9), or characteristics such as the position of a body part, walking speed, and stride length as the style of movement, e.g., (Yamazaki et al., 2005; Wang et al., 2006a; Taylor and Hinton, 2009; Matsubara et al., 2010) (4) (6) (14) (15). A few studies model movement factors by using a more specific characterization scheme than style and content. For instance, Taubert et al. (2012) (22) and Alemi et al. (2015) (29) use factors that represent the categories of emotions or the valence and arousal dimensions of affect, respectively.

The review of the literature reveals two main issues on movement characterization:

1) The definition of the *style* factor varies across the literature, and there is no consensus on what *style* represents. While many studies do not provide any definition of *style*, some refer to it as the quality of movement that changes across the training data. The most viable definition of *style* is used by Brand and Hertzmann (2000) (1) as the variations of the same movement type.

2) Simply using the two dimensions of content and style as the influential factors is insufficient in describing the multifaceted nature of agent movement. A framework with a broader range of dimensions is required to distinguish adequately between various movement qualities, and to better connect those qualities to the internal state of the agent. Only a few studies have addressed associating the controlling factors with the internal state of an agent, e.g., (Herzog et al., 2008; Matsubara et al., 2010; Taubert et al., 2012; Alemi et al., 2015; Starke et al., 2019; Wang et al., 2019) (13) (15) (22) (29) (43) (44).

In the next section, we make the case for a characterization framework tailored for the integration with an agent model that addresses the above shortcomings.

2.3.2 Movement Characterization for Agents

When describing the internal state of an agent and its surrounding environment, we can often put things into multiple semantic dimensions. As the agent movement is influenced by these factors, we can also describe the qualities of its movement across multiple semantic dimensions. The way that these dimensions are laid out and incorporated into a movement generation system plays an important role the application of the movement generation model. For example, movement generation for video games would benefit from a high-level interface that corresponds directly to an agent model and generates the movement according to the changes in the agent's internals. Note that, although inspired by humans,

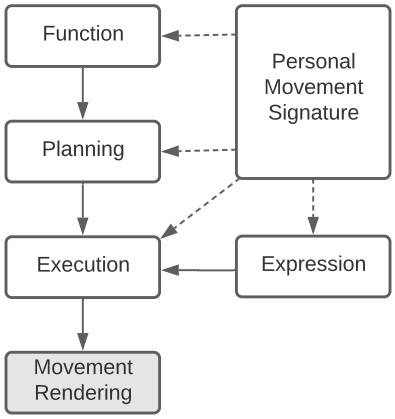


Figure 2.3: The multi-layer framework to describe the movement of virtual animated agents. Solid arrows represent direct influence of one dimension on another while dashed arrows represent indirect influences.

such framework may not exactly mirror how human internal state and movement work together.

Here, we present a framework for characterizing movement that is tailored for integration with virtual agent models. We use this framework throughout this paper to provide a coherent analysis of the literature. The proposed framework consists of six semantic dimensions that characterize an agent's movement : *function*, *planning*, *execution*, *expression*, *personal movement signature*, and *movement rendering* (Fig. 2.3):

Function According to the goal-oriented behaviour of an agent and at the cognitive level, the functional dimension of movement corresponds to the task that the agent is performing through its movement: e.g., reaching the destination from its current location or picking up an object from the table. The functional dimension is perfunctory and does not communicate expression (Cruz-Garza et al., 2014). Note that the function may not always explicitly be present in some movements, such as dancing or abstract movements (Karg et al., 2013).

In modelling the functional dimension of movement in generative models, the common practice is to build a different model for each function. Upon generation, controlling the function of the movement is done by selecting and switching between the available models. A more challenging approach is to build a single model that is capable of generating a variety of functions, e.g., (Li et al., 2002; Qu et al., 2008; Starke et al., 2019, 2020) ③ ⑫ ㊸ ㊼.

Planning The planning dimension is concerned with the sequencing of the fully-body movement and limb motions in order to realize a desired task at specific point in

time and space. For example, moving between two points in a room requires planning the movement in a way to avoid any obstacles, or catching a moving ball, which requires the hands to be at the right position at the right time.

Different sequencing and timing of movements used for planning are often implemented by modelling parameterized movements. For example, a model can be parameterized by the position of end-effectors (Wang et al., 2006b; Herzog et al., 2008; Herzog and Krüger, 2009; Lee et al., 2018) (7) (13) (40), the trajectory of end-effectors (Matsubara et al., 2010) (15), or the location of the agent (Alemi and Pasquier, 2017; Holden et al., 2017; Lee et al., 2018) (36) (39) (40). The exact timing of the movements can also be parameterized by modelling the phase of the movements (Holden et al., 2017) (39).

Execution This dimension encapsulates the patterns that result from the coordinated motion of individual limbs in order to realize the higher-level dimensions of movement in the physical or virtual world. For example, walking (the function) is executed through a locomotion pattern. In other words, the patterns that are defined across the execution dimension act as templates for realizing function and plan variations.

During the training process, the main task of the learning algorithm is to learn the underlying patterns that produce the movements. Any machine learning model would learn one or more execution templates. In addition, some models learn how these templates are modulated by other dimensions, such as expression or planning.

Expression The *expressive* dimension refers to the exhibition of affect through body movement, including the emotions and mood. The expression can be understood as modulation of the execution pattern. For example, for most people, walking while being angry looks different from walking the same path while enjoying it.

The expression in movement can be described using a variety of representations, such as Laban Movement Analysis (LMA) (Bartenieff and Lewis, 1980) for describing movement qualities, or categorical and dimensional representations of affect for emotions (Karg et al., 2013). Controlling the expressivity of the movement is done by learning separate models for each state, e.g., (Tilmanne et al., 2014) (27), or learning movements that are parameterized by factors describing the expressivity e.g., (Taubert et al., 2011; Alemi et al., 2015) (22) (29).

Personal Movement Signature This dimension encapsulates the qualities across the other dimensions that together make the movements of one individual distinguishable from the movements of others.

Movement Rendering At the lowest semantical level, the positions of the individual body limbs are manipulated through space and time, as defined by the execution patterns and modulated by the expressive factors. Movement rendering occurs in the pose space, in which we only deal with the configuration of body parts.

Now that we have discussed how to provide a high-level, semantical representation of movement, we look at how to capture and represent movement at a lower level in the next section.

2.4 Movement Data

Data-driven and statistical movement generation systems do not incorporate any explicit prior knowledge of human movement into their models. As a result, the collection of the movement data that are used in creating such models plays an important role in the generative capabilities and of them.

Table 2.2 summarizes the characteristics of the training data used in the reviewed works. There are a number of choices involved in acquiring a training data set for movement generation, including the type of sensors by which the movement is captured, the way movement data is represented numerically, whether annotations and labels are needed, and the number of human subjects that are available in the data. In addition, there are a number of processing operations that are often performed on the raw data to make them more suitable for a particular machine learning model. In this section, we discuss and review each of these aspects with respect to common practices as reported in the literature. We also include the review of the freely available movement databases that can be used for movement generation.

2.4.1 Capturing Human Movement

A number of sensor systems are used to capture the movements of human actors. Depending on the application, one or more sensor system might be used to capture movement data. These systems vary based on the areas of the body they capture, such as hand movements, full-body movements, expansion of the lungs through breathing, and muscle contractions among others. They also capture different quantities such as position, acceleration, biometrics, energy, etc. Other factors such as the setup requirements (e.g., indoors, outdoors, capturing volume, mobility), the precision and reliability of the measurements, and the sampling rate play a role in choosing the sensor system.

The quantities that sensors capture are summarized in the following categories:

- Joint positions and rotations: motion capture systems
- Joint acceleration and orientation: accelerometer and gyroscope
- Biometric features: electromyography, electroencephalography, breath, heart rate
- Location of the body: Radio Frequency ID (RFID), Global Positioning System (GPS), and Mobile Networks

Throughout the rest of the paper, we focus on motion capture data.

Table 2.2: The training data used in the reviewed models

	Ref	Data Source	Capturing & Input Format	Content	Pre-processing & Feature Extraction
①	(Brand and Hertzmann, 2000)	Various sources / *Unspecified*	Mocap Markers: 20 Framerate (FPS): 60 Raw Dimensions: *Unspecified* Features Dimensions: reduced to < 10 with PCA	- Locomotion - Amateur and professional ballet moves and modern dance *Unspecified number of subjects*	- Reducing dimensions using PCA
②	(Tanco and Hilton, 2000)	Their own (not shared)	FPS: *unspecified* Raw Dimensions: 70 Features Dimensions: reduced to 15 with PCA	- Standing up from the floor - Walking - Running (465 frames total) Single-subject	- Aligning data to be invariant to orientation - Converting the rotations to the angle-axis representation adapted from (Pennec and Thirion, 1997) - Reducing dimensions using PCA
③	(Li et al., 2002)	Their own (not shared)	FPS: 60 Features Dimensions: 60	- Disco dance (49,800 frames) Single-subject	- Converting the rotations to exponential maps
④	(Yamazaki et al., 2005)	Their own (not shared)	Markers: 19 Raw Dimensions: 60 Features Dimensions: 180 FPS: 33	- Walking, varied pace and stride length, 66 sequences Single-subject	- Manually labelling each frame with motion primitives - Calculating 1st and 2nd derivatives of the raw rotations - Calculating the walking pace and stride length
⑤	(Wang et al., 2005)	*Unspecified*	FPS: *unspecified* Features Dimensions: *unspecified*	- Walk (191 frames) - Chop tree (700 frames) - Ballet walk (146 frames) - Ballet roll (169 frames) - Disco (600 frames) - Complexer disco (600 frames) *Unspecified number of subjects*	- Segmenting the elementary behaviour - Converting the rotations to exponential maps - Calculating the first-order derivatives
⑥	(Wang et al., 2006b)	Their own (not shared)	Markers: 19 Features Dimensions: 120 FPS: 33.3	- Normal walking of a male actor - Cat walk of a female character with right arm raised Two subjects	- Converting the rotations to exponential maps
⑦	(Wang et al., 2006a)	Their own (not shared)	FPS: 66 Features Dimensions: *unspecified*	- Boxing (3 minutes) Single-subject	- Manually labeling the motion - Calculating the style values
⑧	(Taylor et al., 2006)	CMU (CMU, 2019), (Hsu et al., 2005)	FPS: 30 Markers: 30 (CMU), 17 ((Hsu et al., 2005)) Features Dimensions: 62 (CMU), 49 ((Hsu et al., 2005))	- Waling and running - Stylistic walks Single-subject	- Converting the rotations to exponential maps - Feature standardization - Removing constant zero dimensions
⑨	(Wang et al., 2007)	CMU (CMU, 2019)	FPS: 30 Features Dimensions: 89	- Locomotion Three subjects	- Calculating rotation and translation velocities - Converting the rotations of joints with 3 DOFs and the global orientation to exponential maps - Labelling the gait type
⑩	(Wang et al., 2008)	CMU (CMU, 2019)	FPS: 60 & 30 Features Dimensions: 50	- Single-subject walk (260 frames) - Four-subjects walk (1146 frames) - Golf club swing (four samples, 1015 frames)	*N/A*

Table 2.2: The training data used in the reviewed models - *Continued*

	Ref	Data Source	Capturing & Input Format	Content	Pre-processing & Feature Extraction
(11)	(Lin et al., 2008)	Their own (not shared)	Features Dimensions: 21 positions (5 arm joints + initial positon + final postion in 3D)	- Lifting arm movements (four repetitions)	*N/A*
(12)	(Qu et al., 2008)	CMU (CMU, 2019)	FPS: 60 Raw Dimensions: *unspecified* Features Dimensions: 5 & 8	- Boxing, Indian dancing Single-subject	*N/A* - Reducing dimensions with Isomap
(13)	(Herzog et al., 2008; Herzog and Krüger, 2009)	Their own (not shared)	Markers: 7 Dimensions: *unspecified* FPS: *unspecified* Using the 3D position of markers instead of rotations	- Grasping, pointing Single-subject	*N/A*
(14)	(Taylor and Hinton, 2009)	CMU (CMU, 2019), (Hsu et al., 2005) (quantitative analysis), Their own (not shared) (multiple style variables)	FPS: 60 Markers: 30 (CMU), 17 ((Hsu et al., 2005)) Features Dimensions: 62 (CMU), 49 ((Hsu et al., 2005))	- Stylized walks - Multiple style variables: cross-product of (slow, normal, fast) speed and (short, normal, long) stride length. 6000 frames. - Quantitative analysis: seven types of walking, each at three different speeds Single-subject	- Converting the rotations to exponential maps - Feature Standardization - Removing constant zero dimensions - Labelling the data
(15)	(Matsubara et al., 2010)	Their own (not shared)	FPS: *unspecified* Features Dimensions: 12 Using markers positions	- Table tennis swings (15 sequences) - Reaching (15 sequences) Single-subject	*N/A*
(16)	(Tilmanne and Dutoit, 2010)	Their own (not shared)	FPS: 30 Features Dimensions: 54	- Stylized walks (247 cycles) Single-subject	- Removing the root translation - Re-sampling the data to have a fixed length - Converting the rotations to quaternions for re-sampling - Converting the rotations exponential maps for PCA
(17)	(Wei et al., 2011)	*Unspecified*	FPS: *Unspecified* Features Dimension: 19, 22, and 19	- Waling variations: step sizes, turning angles, walking speeds, and walking slopes - Stylized walking - Locomotion *Unspecified number of subjects*	- Reducing dimensions using PCA
(18)	(Liu et al., 2011)	Their own (not shared)	FPS: *unspecified* Features Dimensions: *unspecified*	- Sideways stepping (75 sequences) - Reaching (70 sequences) - Stride over obstructions (78 sequences) 12 subjects	- Time warping all movement to a reference movement - Manually specifying key-frames for each sequence - Reducing dimensions using PCA
(19)	(Kulić et al., 2011)	Their own (not shared)	FPS: 30 Markers: 34 Features Dimensions: 90 and 120 Using markers positions	- Walking, squatting, kicking, raising an arm - Single-subject	*N/A*
(20)	(Chiu and Marsella, 2011b)	Data from Human Sensitivity study (Ennis et al., 2010)	FPS: *unspecified* Features Dimensions of the arm joints: 21	- Debate conversations (1140 frames) Three subjects	- Converting the rotations to exponential maps

Table 2.2: The training data used in the reviewed models - *Continued*

	Ref	Data Source	Capturing & Input Format	Content	Pre-processing & Feature Extraction
(21)	(Chiu and Marsella, 2011a)	CMU (CMU, 2019)	FPS: *unspecified* Features Dimensions: 96	- Stylized walks Single-subject	- Removing the global translation - Converting the rotations to exponential maps
(22)	(Taubert et al., 2011, 2012)	Their own (not shared)	FPS: *unspecified* Features Dimensions: 38 and 159	- Handshakes with four emotions Two subjects	*N/A*
(23)	(Tilmanne et al., 2012)	Proprietary: Mockey (Tilmanne and Dutoit, 2010) eNTERFACE'08 3D (Tilmanne et al., 2008)	Inertial sensors: 18 Raw dimensions: 54 Dimensions with speed and acceleration: 162 FPS: 60 (Mockey), 30 (eNTERFACE'08 3D)	- Mockey: Walks in 11 styles: proud, decided, sad, top model, drunk, cool, afraid, tiptoeing, heavy, in a hurry, and manly. - eNTERFACE'08 3D: Neutral walking sequences of 41 actors	- Aligning the direction of all the walking sequences. - Manually segmenting the sequences into left and right steps - Converting to exponential maps - Calculating the speed and acceleration of the rotations
(24)	(Min and Chai, 2012)	Their own (not shared)	Markers: *Unspecified* Features Dimensions: *Unspecified* FPS: *Unspecified*	- Standing, walking, running, two-feet jumping, stepping-stone jumping, sitting down, standing up, climbing up, climbing down, left punching, right punching, picking, placing, kneeling down, kneeling up, backward walking - Transitions between the above	- Extracting keyframes - Segmentation - Segment registration for each primitive - Functional decomposition of each segment
(25)	Levine et al. (2012)	Their own	FPS: 30	- Locomotion - Karate	- Calculate joint velocities
(26)	(Samadani et al., 2013)	Data from (Samadani et al., 2011) II	FPS: 84 Features Dimensions: 54	- Closing and opening the hand with happy, sad, and angry emotions	- Re-sampling the data to be aligned and have a fixed length - Modelling the data with Basis Function Expansion method
(27)	(Tilmanne et al., 2014)	Mockey Database (Tilmanne and Dutoit, 2010)	FPS: 30 Markers: 34 Features Dimensions: 54	- Stylized walks Single-subject	- Automatically annotating frames with left and right steps - Converting the rotations to exponential maps
(28)	(Fragkiadaki et al., 2015)	H3.6M (Ionescu et al., 2014)	Markers: 30 FPS: 50 Features Dimensions: 54	- Walking, eating, smoking	- Converting the rotations to exponential maps - Calculating the speed of the rotation and global translation - Feature standardization
(29)	(Alemi et al., 2015)	Their own (shared) (MovingStories, 2019)	FPS: 30 Markers: 53 Features Dimension: 52	- Affect-expressive walks (36,000 frames) Two subjects	- Converting the rotations to exponential maps - Feature standardization - Removing constant zero dimensions - Labelling each training sequence with the valence and arousal values
(30)	Jain et al. (2015)	Human3.6m	Markers: 30 FPS: 50 Exponential Maps	- Walking, eating, smoking, and having a discussion	*N/A*
(31)	(Crnkovic-Friis and Crnkovic-Friis, 2016)	Their own (not shared)	FPS: 30 Features Dimensions: 75 Using 3D positions	- Contemporary Dance	*N/A*

Table 2.2: The training data used in the reviewed models - *Continued*

	Ref	Data Source	Capturing & Input Format	Content	Pre-processing & Feature Extraction
(32)	(Holden et al., 2016)	CMU (CMU, 2019) + HDM05 (Müller et al., 2007) + (Ofli et al., 2013) + (Xia et al., 2015)	FPS: 60 Features Dimensions: 70	All of the contents of the used databases, retargetted into a uniform skeleton, resulting in around six million frames (at 60 FPS)	- Converting the joint rotations to posittions with respect to a body-centric coordinate system - Calculating the body orientation and global velocities - Applying Gaussian filters to reduce noise - Finding foot contact points - Feature standardization - Segmenting the data into overlapping windows
(33)	(Wang and Artières, 2017)	Emilya Dataset (Fourati and Pelachaud, 2014)	FPS: 120 Features Dimensions: *Unspecified*	- 12 performers - 8 activities - 8 emotions	*N/A*
(34)	(Herrmann et al., 2017)	Their own (not shared)	Joints: 20 FPS: *Unspecified* Features Dimensions: 79	- Walking - Picking - Placing - Screwing	- Converting the rotations to quaternions - Segmentation - Temporal and spatial alignment - Feature standardization - Smoothing
(35)	(Martinez et al., 2017)	Human 3.6M	Markers: 30 FPS: 50 Features Dimensions: 54	- Seven performers - Walking, smoking, engaging in a discussion, taking pictures, and talking on the phone - Two different trials for each performer/actions	- Converting the rotations to exponential maps
(36)	(Alemi and Pasquier, 2017)	Their own (shared) (MovingStories, 2019)	FPS: 30 Markers: 53 Features Dimension: 52	- Affect-expressive walks (36,000 frames) - Two subjects	- Converting the rotations to exponential maps - Feature standardization - Removing constant zero dimensions - Labelling the valence and arousal values for each training sequence - Labelling each frame with the orientation of the body
(37)	(Alemi et al., 2017)	Their own (shared) (MovingStories, 2019)	FPS: 30 Markers: 53 Features Dimension: 52	- Dance movements - Audio features	- Converting the rotations to exponential maps - Feature standardization - Removing constant zero dimensions - Labelling each frame with corresponding audio features
(38)	Li et al. (2017)	CMU (CMU, 2019)	FPS: 60 Joints: 57	- Martial Arts - Indian dance - Indian/Salsa Hybrid - Walking	- Calculate root velocity - Calculate joint positions relative to the root position in a world-coordinate system
(39)	Holden et al. (2017)	Their own	FPS: 60 Joints: 30 Joint positions and velocities in a root-centric coordinate system	- Locomotion, including walking, jogging, running, jumping over obstacles, at different speeds	- Annotate phase data - Annotate semantic labels of gait - Calculate body trajectory - Calculate the height information of the train
(40)	Lee et al. (2018)	Their own	FPS: 30 Both joint rotations and positions	- Basketball - Tennis - Locomotion including walking turning, side stepping, and backward stepping	- Annotate the contact info (foot and floor, hand and ball)

Table 2.2: The training data used in the reviewed models - *Continued*

	Ref	Data Source	Capturing & Input Format	Content	Pre-processing & Feature Extraction
(41)	Pavllo et al. (2018)	- Human3.6m for short-term tasks - Locomotion from CMU (CMU, 2019) , HDM05 (Müller et al., 2007), Berkley MHAD, and data from (Xia et al., 2015) for controlled, long-term generation	FPS: 25 for Human3.6m 30 the rest Joints: 26	- All of the movements in Human3.6 (no control) - Locomotion	- Convert joint rotations to quaternions - Extract trajectories and locomotion control parameters
(42)	Li et al. (2018)	Human3.6m and CMU (CMU, 2019)	Joints: 32 Exponential Maps	- Walking, eating, smoking, and having a discussion from Human3.6m - Various movements from CMU	- Set the global position and orientation to zero
(43)	Starke et al. (2019)	Their own	Bones: 23	- Walking, running, sitting, opening doors, picking up, carrying, and putting down boxes.	- Contact and phase extraction - Goal and action annotation
(44)	Wang et al. (2019)	Their own and CMU (CMU, 2019) (pre-training)	FPS: 30 Joint angles	- Walking - Running	- Calculate the root's translational velocity - Calculate the root's angular velocity along the vertical axis
(45)	Pettee et al. (2019)	Their own	Joints: 53 FPS: 35 Size: 30 minutes Joint Positions	- Improvisational Dance	- Annotate phase data - Annotate semantic labels of gait - Calculate body trajectory - Calculate the height information of the train
(46)	Sun et al. (2021)	Their own (Youtube-Dance3D)	Joints: 24 Exponential Maps	- Dancing	*N/A*
(47)	Starke et al. (2020)	Their own	Bones: 26 FPS: 30	- Basketball moves (standing, moving, dribbling, holding, shooting)	- Contact and phase extraction - Goal and action annotation
(48)	Ling et al. (2020)	Their own	FPS: 30	- Walking, running, turning, stopping, resting	*N/A*

Motion Capture

Motion Capture (Mocap) is a popular approach for recording movement and is widely used in the movie, video game, sports, and health care industries. Mocap systems use marker-based or marker-less techniques to capture the trajectories of body limbs in a 3D coordinate system. Mocap markers can be acoustic, inertial, magnetic, reflective, or a combination of these. Marker-less systems use computer vision techniques to track the optical flow of the pixels in a 2D video stream of movement (RGB and infrared), as used in Microsoft Kinect. Special motion capture systems for capturing the movements of hands and fingers can be worn like a glove, e.g., (Lu et al., 2009).

Motion capture systems are often used to capture whole body movements. However, it is also commonly used for capturing detailed limb movements, e.g., (Samadani et al., 2011), as well as facial expression.

Regardless of the capturing techniques, the trajectories of the markers or pixels are often mapped to a virtual skeleton, defined by a hierarchy of joint angle rotations that ensures that the body limbs have fixed lengths. While most of the approaches use joint rotations, the trajectories are also directly used for modelling movement (Kulić et al., 2011; Crnkovic-Friis and Crnkovic-Friis, 2016) (8) (31).

Motion Capture Data Representation

Each frame of motion capture data consists of a root node which defines the body's *absolute* position and orientation with respect to a global Cartesian coordinate system, and a set of nodes each representing a joint's or bone's orientation. Each node, depending on what part of the body it corresponds to, can be represented by 1 (e.g. knee), 2 (e.g. wrist), or 3 (e.g. arm) parameters, also called Degrees Of Freedom or DOF. Each parameter or DOF describes the rotation of the joint/bone along one of the axes of a 3D coordinate system *relative* to its parent joint/bone. These parameters constitute the input of the machine learning pipelines.

There are different parameterizations to represent the joint angle rotations. None of these representations is perfect and depending on the application, one might be chosen over another. One approach to represent rotations is to use rotation matrices. A 3D transformation represented by a rotation matrix requires a 3×3 matrix, consisting of nine values. Because of the larger number of values and the need for imposing constraints on various operations, it is not efficient to use rotation matrices for most of the applications. Other common parameterizations are discussed below.

Euler Angles is one of the most common representations for rotations in movement data and describes a 3D rotation using three separate angles one for each axis. The rotations around each axis is done in a predetermined order. Euler angles are widely used in animation industry as they are easy to understand and manipulated by humans. However, Euler angles

suffer from discontinuities and are susceptible to the loss of a degree of freedom when two of the three axes of a 3D rotation are aligned (also known as a *gimbal lock*). Thus, few computational models use Euler angles in practice (Wang et al., 2007) (9).

Quaternions are 4D unit vectors representing 3D rotations. They do not suffer from gimbal locks and discontinuities and can be interpolated using the SLERP algorithm (Shoemake and Shoemake, 1985). Downsides of the quaternions are the extra dimension (compared to 3-dimensional vectors), the need to enforce the constraint that the vector has a unit length, and that quaternion representations are not unique.

Exponential map is another technique that is applied to motion capture representations. "The exponential map maps a vector in $\mathbb{R}^3$ describing the axis and magnitude of a three DOF rotation to the corresponding rotation" (Grassia, 1998). Exponential map allows for interpolation, is not susceptible to gimbal locks when used for modelling human movement, and does not require constraints to be imposed.

Regardless of which rotation parameterization is used to train the model, they all have the benefit of ensuring that the body limbs have fixed lengths. In addition, rotation parameterizations can be converted to one another. As a result, the output can be converted to one of the standard mocap formats that are used by animation software and drive an animated character.

Although *joint positions* are not suitable for animation, machine learning models can benefit from representing movements using positions. Compared to rotation representations, positions are less ambiguous to the learning algorithms. In addition to using positions to directly train models, Pavllo et al. (2018) (41) suggest training models on joint rotations to take advantage of the compatibility with the animation pipelines, but to calculate the loss function of neural networks based on the joint positions to take advantage of the less ambiguous position representation.

2.4.2 Training Data for Movement Modelling

In the following, we discuss those aspects of modelling movement data that are relevant to creating generative movement models.

Sampling Rate

The sampling rate (frame rate) of the data represents how many measurements are recorded by the sensors in a window of time. In order to capture fast-paced movements, a high sampling rate is needed to produce a smooth recorded movement. The sampling rate of the training data might be adjusted to comply with the space and computational complexities of the statistical models, as well as to combine data from different sources that have different sampling rates. While the original data might be recorded in higher frame rates (e.g., 120HZ), most approaches down-sample the data (e.g., to 30HZ) to reduce the size of

the training dataset. Most of the motion capture formats use a fixed sampling rate when recording the data.

Pre-Processing & Feature Extraction

A generative system might require the raw mocap data to go through a series of processes to make the data usable for the learning algorithm. Some common processes include, but not limited to:

Data Representation - It is common to change the data representation to one of the representations mentioned in Section 2.4.1 before feeding them to the machine learning model. In addition, approaches that use functional statistics transform the data using basis functions (Samadani et al., 2011).

Segmentation - Some approaches use segmentation to break down long sequences or to organize the system into hierarchical structures. As discussed in more details in Section 2.5, the segmentation can be done based on identification of elementary movements or based on choosing windows of fixed length.

Alignment and Length Normalization - Some approaches require the training data to have fixed lengths in such a way that similar movements (e.g., each walking cycle) are aligned. The alignment and resampling is done using the SLERP algorithm (Shoemake and Shoemake, 1985) or using piece-wise linear re-sampling.

Rotational or Positional Velocity and Acceleration - Some studies calculate the velocity and acceleration of each DOF of the movement and add the extra features to the training data, as in (Wang et al., 2005; Yamazaki et al., 2005; Wang et al., 2007; Tilmanne et al., 2012; Starke et al., 2019, 2020) (5) (4) (9) (23) (43) (47).

Derived Movement Features - It is also possible to derive other features from the movement data using analytical formulas. For example, the stride length is directly extracted from the data and used as labels to annotate the data (Yamazaki et al., 2005) (4).

Dimensionality Reduction - The dimensionality is another important characteristic of training data. The number of dimensions of each frame (i.e., the feature vector) is determined by the type of the rotation parameterization, the number of data points corresponding to the markers, joints, or bones, and any extra movement features that might be added to the feature vector such as the velocity or the acceleration of the joint rotations. As the dimensionality of the data increases, it can be more difficult for some machine learning models to learn the underlying patterns. Often many of the dimensions of the data do not carry much information about the underlying patterns and therefore can be removed from the training data without loosing much information. As a result, dimensionality reduction techniques, such as Principal Component Analysis (PCA) (Jolliffe, 2002), are applied to the data to identify the dimensions that cause the most variations in the data and eliminate the ones that do not carry much information. A reduced feature vector is then used for training the model, e.g., (Brand and Hertzmann, 2000; Liu et al., 2011; Wei et al., 2011) (1) (18) (17).

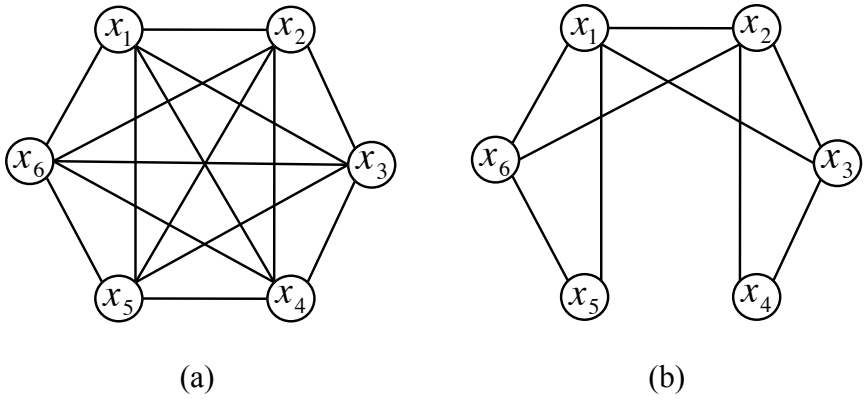


Figure 2.4: Comparison of (a) a Gaussian distribution with a full covariance matrix and (b) a DTG distribution with three components (the triangles). Each node represents a dimension of the data. Edges represent the dependencies between the dimensions. The DTG components are $\{x_1, x_5, x_6\}$, $\{x_1, x_2, x_3\}$, and $\{x_2, x_3, x_4\}$.

Feature learning - Instead of solely using the raw movement data or the features that are derived from analytical approaches (also known as feature engineering), one can derive features through an unsupervised learning processes. In such learning process, a machine learning algorithm is used to learn a new representation of the data that could posses characteristics that are more efficient for learning a generative model that the raw data or analytical features. This is more common in deep learning applications in which a neural network is first trained on a large amount of training data in an unsupervised way, and then a second model is trained on a possibly smaller dataset that is used directly for the generation.

Normalization - For approaches that use artificial neural networks, the training process converges more efficiently when the training data vectors are normalized to have zero mean and unit standard deviation. The normalization is often the latest stage of the preprocessing and is applied to each dimension of the data independently.

Probability Distribution

The multivariate Gaussian distribution is the most commonly used probability distribution to model the multi-dimensional movement data. In addition to the Gaussian distribution, Wang et al. (2005) argue that the high-dimensional human movement data can be modelled more efficiently with the assumption that each dimension only depends on two other dimensions (Song et al., 2003), and can be represented by a decomposable triangulated graph (DTG), as shown in Fig. 2.4(b)). Each triangle in the DTG is modelled by a 2-dimensional or 3-dimensional Gaussian distribution. DTGs are applied in a few movement generation studies, e.g., (Wang et al., 2005, 2006b) ⑤ ⑦. Refer to the article by Song et al. (2003)

for more information and discussion of the differences between the Gaussian and DTG distributions.

Data Annotations

Data annotation have two applications: 1) to segment the training data, and 2) to train the model to control the movement given a description. In the former, the annotations are only used to segment the training data before the training process and the machine learning model does not use the annotations. In the latter, the annotations are directly used to train the model, which allows the model to learn the associations between the given descriptors, and the underlying mechanisms that generate the movement patterns.

The annotations are often marked manually by human observers. They can be discrete to represent categorical data such as the mover's identity or gait type, or continuous to represent real-valued measurements such as stride length or hand position. Depending on the semantics of the descriptors and the learning mechanisms, the annotations can be associated to whole sequences of the training data for to each frame individually.

Some of the descriptors that are used in the literature include *the right and left steps* of a walking cycle, e.g., (Yamazaki et al., 2005; Tilmanne et al., 2012, 2014) (4) (23) (27); the body height and the distance of a punching action (Wang et al., 2006a) (6); the affective qualities of the movement, in terms of categorical emotions, e.g., (Samadani et al., 2013) (26), or in terms of the valence and arousal dimensions, e.g., (Alemi et al., 2015) (29); the walking speed and stride length (Taylor and Hinton, 2009; Yamazaki et al., 2005) (14) (4); the facing direction (Alemi and Pasquier, 2017; Pavllo et al., 2018) (37) (41); or arbitrary class labels, e.g., the gait styles during walking (Taylor and Hinton, 2009) (14).

Another group of annotations are with respect to the contact information between the mover, the ground it is moving on, objects the mover might interact with, and other movers in the environment, e.g., (Holden et al., 2017; Lee et al., 2018; Starke et al., 2019, 2020) (39) (40) (43) (47). Such annotations indicates the frames that a contact occurs.

Number of Subjects

The modulations and variations of human movement are affected by the personal movement signature of the performer. Each person has a different movement signature, which is influences by their genetics, habits, attitudes, values, and life history (Studd and Cox, 2013). To learn a generalized model of movement, which is invariant to the performer-specific styles while recognizing the "personal factor" of movement, requires training the model using the data from multiple subjects.

The majority of studies use a single subject in their training data, while some studies use two, e.g., (Wang et al., 2006b; Taubert et al., 2011, 2012; Alemi et al., 2015) (7) (22) (22) (29), three subjects, e.g., (Wang et al., 2007; Chiu and Marsella, 2011b) (9) (20), and in

Table 2.3: A summary of the motion capture databases

DB	Purpose	Content and Size	Characterization	Recording Tech	Modalities	Subjects	Redundancy	Annotation
Ohio State University's ACCAD (ACCAD, 2019)	Video Games & Animation	300 sequences - Locomotion - Gestures - Martial Arts	Function with Planning Variations	- Vicon	Marker Data, Skeletal Data	3	One instance per performer	-
AffectMe (AffectMe, 2016)	Study of body posture as an indicator of human affective states, pain, and immersion	Collection of datasets: - Acted emotions - Non-acted affective states in computer game settings	Expression	- Vicon - Gypsy5	-	-	-	Categorical Emotions
CMU Mocap (CMU, 2019)	General Research	2605 sequences in five categories: - Human Interaction - Interaction with Environment - Locomotion - Physical Actitivties & Sports - Situations & Scenarios	Function with Planning and Expression Variations	- Vicon with 41 markers	Marker Data, Skeletal Data, Video, Animation	144 (including duplicates)	Varies for different functions	-
Cologne DB(Cologne DB, 2019)	*Unspecified*	- Locomotion - Arm Gestures - Skydiving, Swimming, Climbing	Expression with variations in Function	26 joints	Marker Data, Skeletal Data	29	Different roles, moods, genders, and actors	-
Dance Motion Capture Database of the University of Cyprus. (DanceDB, 2019)	Digital Archive of Dance	- Greek and Cypriot Dances	Expression	- Phasespace Impulse X2 with 38 markers	Marker Data, Skeletal Data, Video	3	-	-
HDM05 (Müller et al., 2007)	General Research	3 hours of motion captures in 70 different classes - Locomotion - Grabbing and Depositing - Sports - Sitting and Lying Down - Dance	Function with variations in Planning and Performer	- Vicon with 40 markers and 24 joints	Marker Data, Skeletal Data	5	10 to 50 instances of the same function performed by different subjects	-
HumanEva-I (Sigal and Black, 2006)	Human movement and pose estimation from video data	- Walking - Jogging - Gestures - Throw/Catch - Combinations of the above	Function	- ViconPeak	Synchronized Marker Data and Video	4	2 instances per function	-
IEMOCAP (Busso et al., 2008)	- Recognition and Analysis of Emotional Expression - Analysis of Human Dyadic Interactions - Design of Emotion-Sensitive Human Computer Interfaces and Virtual Agents	12 hours - Facial expression and head and hand movements + the audio recordings of the conversations	Function with variations in Expression	- Vicon with 53 facial markers, 3 markers for each hand, and 2 markers on the head	Facial Mocap, Head Movement and Orientation, Video, Speech, Dialog Transcript	10	-	Categorical and dimensional emotions
MoDa (MovingStories, 2019)	- Studying movement and meaning based on Laban Movement Analysis - Creating motion graphs - Affect-expressive movement generation	Collection of Datasets: - Affect-Expressive Motion Graph - LMA Basic Effort Actions - Knocking and Direction Gestures with Variation form LMA - Grooving	Function, Expression, Planning	- Vicon with 53 markers and 26 joints	Marker Data, Skeletal Data, Video, (Audio and Physiological, for some datasets)	Varies (1 to 3 performers)	1 to 4 repeatitions across function and expression	Laban Movement Analysis, Dimensional Emotions

Table 2.3: A summary of the motion capture databases - *Continued*

DB	Purpose	Content and Size	Characterization	Recording Tech	Modalities	Subjects	Redundancy	Annotation
NUS DB (NUSDB, 2019)	*Unspecified*	- Locomotion - Interaction with Obstacles - Martial Arts - Dance - Yoga	Function with planning variations	- Vicon	Marker Data, Skeletal Data	8	-	-
UPenn DB (U Penn DB, 2016)	- Multi-Actor behaviours - Diverse personalities - The effects of posture and dynamics on the perception of emotion - Study human fatigue	Collection of multimodal datasets: - Walking - Emotional Actions - Emotional Body Language - Exercise	Function and Expression	*Unspecified*	Marker Data, Skeletal Data, ForcePlate, Biological Data	Varies	-	Categorical Emotions
(Ma et al., 2006)	Study of identity, gender, and emotion perception	4080 sequences - Walking - Knocking - Lifting - Throwing	Function and Expression	Falcon Analog, 35 markers, 15 joints	Marker Data, Skeletal Data	30	5 repetitions	Categorical Emotions
NTU RGB+D (Shahroudy et al., [n.d.])	RGB+D human action recognition	60 action classes within daily actions, health-related actions, and inter-personal actions	Function	- Three Microsoft Kinect V2 devices	Video, Joint Positions	40	2 instances for each actions (two different angles from the camera)	Action labels
Berkeley MHAD (Ofli et al., 2013)	RGB+D human action recognition	11 actions with high dynamics in: - Both upper and lower body - Upper body - Lower body	Function	- Mocap: Impulse - Video: 12 Dragonfly2 cameras - Depth: 2 Microsoft Kinect V2 - Acceleration: 6 three-axis wireless accelerometers on wrists, ankles, and hips	Joint positions, depth, multi-angle video, audio, acceleration	5 female, 7 male	5 repetitions for each action	Action labels
Human 3.6M (Ionescu et al., 2014)	Human pose estimation	15 actions within upper body movement, full-body upright variations, walking variations, sitting on the floor, and miscellaneous movements	Function	- Mocap: Vicom T40 - TOF: Mesa SR4000 - Video: Basler piA1000 - Body Scan: Vitus Smart LC3	RGB video, depth, joint positions, 3D volumetric models of subjects	5 female and 6 male subjects	One repetition	Action labels, body parts (for video)
Mixamo (Adobe, 2020)	Video games and animation	Various video game actions	Function	*unspecified*	- Skeletal Data - 3D Character Models	*unspecified*	-	-

other cases 12 and 41 subjects, e.g., (Liu et al., 2011) (18) and (Tilmanne et al., 2012) (23), respectively.

The number of subjects used in the studies is, to some extent, limited by the availability of the data from multiple actors, performing the same type of movements. This number varies in the publicly available training databases, which is discussed in the following section.

2.4.3 Movement Databases

The majority of the reviewed studies use the data from the Carnegie Mellon University Motion Capture Database (CMU mocap) or have captured their own data. In addition to the CMU database, there are a number of other databases that are publicly available for research purposes and potentially can be used for movement generation. Key details of these movement databases are presented in Table 2.3 and are discussed in the following.

Curation and Purpose

Most of the databases are created and tailored towards a set of particular research questions. For example, the primary goal of the IEMOCAP, University of Glasgow, and AffectMe databases is towards the analysis of emotional expression, while the University of Pennsylvania database is tailored towards modelling multi-actor behaviours. Mixamo provides animation clips tailored for actions that are common in video games. Databases such as the CMU, ACCAD, and HDM05 provide a wider and more general set of contents that are created to provide freely available motion capture data to the research community for a variety of purposes. MoDa, an open-source movement database, is a repository of multiple databases that address a range of movement-related research questions such as affect-expressive motion graphs, data tailored for Laban Movement Analysis research, and dance and music studies.

Some databases, such as Berkeley MHAD, NTU RGB+D, and Human 3.6M, are created for the research on human movement analysis and action recognition from image and video (RGB) data in the context of every-day human activities. However, they often include reference motion capture data which can be used for training generative movement models.

Content and Size

The size of the presented databases varies extensively both in terms of the length of the content and the diversity of the movements. While some databases provide a relatively large amount of motion captured data (e.g., Human 3.6M, CMU, IEMOCAP, and MoDa), others only have a few sequences (e.g., Cypress DanceDB).

Characterization Diversity

To effectively create systems that model and control variations across different dimensions of movement (as described in Section 2.3), one has to have access to a training dataset that contains the desired variations. The majority of the databases contain movements that vary across the function and expression dimensions, while a fewer number of databases contain variations across the planning dimension.

Recording Technology and Files Formats

The majority of the database use Vicon motion capture systems with reflective markers, some use mocap systems with infrared markers, and a few use inertia-based capturing systems.

All of the databases provide the raw marker data in C3D format (Dainis, 2019), and most provide skeletal data (joint angle rotations) in the form of BVH (BVH, 2019) or AMC (Acclaim, 2013) files. Multi-modal data sets are also available in some databases and provide video, audio, or physiological recordings that accompany the movement.

Capturing Modalities

Most databases provide the raw marker data, as well as the skeletal data. Some also accompany a video recording of the motion capture session for each movement as a reference. Few databases such as MoDa, IEMOCAP, and Berkeley MHAD provide other modalities such as voice, text, facial expression, and physiological measures.

Human Subjects

Every human has a distinct movement signature and style (Studd and Cox, 2013). The more the number of subjects in a training dataset, the better is the model's ability to distinguish between these personal modulations in the data and the underlying patterns that is common among the movements of all performers.

The number of subjects varies from 1 to 144 in the reviewed databases. Note that in some cases, the same movement may not be available for all of the subjects. For example, in the CMU database the same movement is only repeated by very few subjects, rather than the whole 144 movers. On the other hand, databases such as IEMOCAP or MoDa ensure that the same movements are consistently performed by all subjects.

Repetitions and Motor Variation

In most machine learning problems, including learning generative movement models, having more variations of the data increases the robustness of the model towards the variations that the model faces in real-world applications and avoids overfitting the model to a limited set of input. While many databases provide no or very few repetitions, databases such as

HDM05, MoDa, and the University of Glasgow provide multiple repetitions of the same movement.

Annotations

In the reviewed databases, annotations mostly include the categorical emotions (as in IEMOCAP, University of Pennsylvania, University of Glasgow, and AffectMe) and the dimensional affect representations (as in IEMOCAP and MoDa). Databases in MoDa also include annotations based on the Laban Movement Analysis (Bartenieff and Lewis, 1980).

2.5 Learning and Generation

In this section, we analyze the learning and generation methods that are applied to the motion capture data. We organize our analysis based on the machine learning families, namely dimensionality reduction techniques, Gaussian processes, Hidden Markov Models, artificial neural networks, as well as a few other machine learning approaches. A summary of the different machine learning approaches is shown in Table 2.4.

2.5.1 Dimensionality Reduction

A dimensionality reduction (DR) model learns a mapping that transforms the data to a lower-dimensional representation while preserving as much information about the original data as possible. As we saw in Section 2.4.2, DR techniques are applied to the training data before being fed into another machine learning model (e.g., Hidden Markov Models) to reduce the memory usage during the training process and increase the learning speed.

Here, we look at the application of DR techniques in directly generating movement. We should note that by DR we only refer to the techniques specifically designed for dimensionality reduction as DR can also be achieved by other techniques that are not specifically designed for DR. For example, neural networks can be designed to learn a low-dimensional representation of data as in Autoencoders (Section 2.5.4).

The main idea behind using a DR technique for movement generation is to reverse the direction of the mapping between the high-dimensional data and the resulting low-dimensional DR space. Once a DR model is trained, one can choose a point in the DR space, and map the point back into the high-dimensional space to generate a new sample. Note that this transformation is lossy, meaning that the reconstructed samples are only to a certain degree similar to the original data.

Using DR techniques for movement generation has some advantages. Manipulating a variable in a lower-dimensional space is much more convenient than manipulating a high-dimensional variable with many inter-dependent dimensions. This property can be used to use the low-dimensional space as the control space of the movement generation. In addition, the dimensions in the DR space could correspond to meaningful qualities of the data in some

Table 2.4: Machine Learning Methods for Movement Learning and Generation

Machine Learning Family		Model	Details	Factorization Technique	Remarks
Dimensionality Reduction	(12)	Isomap embedding (Qu et al., 2008)	Modelling dynamics with LDS		
	(16)	Principal Component Analysis (PCA) (Tilmanne and Dutoit, 2010)		Principal Components	
	(24)	Functional PCA + Gaussan Mixture Model + Gaussian Process (Min and Chai, 2012)		Graphs + Optimization	
	(26)	Functional PCA (Samadani et al., 2013)		Principal components + clustering	
	(34)	Functional PCA + Gaussian Mixture Model + Gaussian Process + kMeans Trees (Herrmann et al., 2017)		Graphs + Optimization	
Gaussian Process Models	(9)	Multifactor Gaussian Process Models (Wang et al., 2007)	Dynamic model	Latent space	
	(10)	Gaussian Process Dynamical Models (GPDM) (Wang et al., 2008)		Dynamic model	
	(22)	Gaussian Process Latent Variable Models (GPLVM) (Taubert et al., 2011, 2012)	Modelling dynamics with HMM	Individual models	Modelling two-character handshake
	(25)	Gaussian Process Latent Variable Models (GPLVM) (Levine et al., 2012)	Augmented with a connectivity prior	Building a control policy over the latent space	-
Hidden Markov Models	(1)	Stylistic HMM (Brand and Hertzmann, 2000)	75 states	Parametric Gaussian	Unsupervised learning of movement factors
	(4)	Multiple Regression Hidden Semi-Markov Models (Yamazaki et al., 2005)	5 states	Parametric Gaussian	Modeling walking pace and stride length
	(5)	Hierarchical HMM (Wang et al., 2005)			Hierarchical model - Using DTG

Table 2.4: Machine Learning Methods for Movement Learning and Generation - *Continued*

Machine Learning Family		Model	Details	Factorization Technique	Remarks
Hidden Markov Models	⑦	HMM/Mix-SDTG (Wang et al., 2006b)	4 states	Parametric Gaussian	Using mixture of SDTGs
	⑬	Parametric HMM (Herzog et al., 2008)	20 states	1.Interpolating individual models 2. Parametric Gaussian	
	㉓	Hidden Semi-Markov Models (Tilmanne et al., 2012)	5 states	Average model + Individual stylized models + Transformation algorithms	Learns a neutral walking model which can be adapted to different styles
Artificial Neural Networks	⑥	Self-Organizing Mixture Network or SOMN (Wang et al., 2006a)	1 layer	Parametric Gaussian	
	⑧	Conditional RBM (CRBM) (Taylor et al., 2006)	1 and 2 layers		Unsupervised learning
	⑪	Feed-Forward Network (Lin et al., 2008)	1 layer	Regression	Lifting movement
	⑭	Factored CRBM (Taylor and Hinton, 2009)	1 layer	Controlling network weights	Supervised learning and control
	⑳	Hierarchical FCRBM (Chiu and Marsella, 2011a)	2 layers	Controlling network weights	Interpolation
	㉑	Hierarchical FCRBM (Chiu and Marsella, 2011b)	2 layers	Controlling network weights	Gestures controlled by audio
	㉘	Encoder-Recurrent-Decoder (Fragkiadaki et al., 2015)	1 encoder, 2 recurrent, and 1 decoder layers		Learning the representation of posture using fully-connected layers at the same time as training the recurrent layers

Table 2.4: Machine Learning Methods for Movement Learning and Generation - *Continued*

Machine Learning Family		Model	Details	Factorization Technique	Remarks
Artificial Neural Networks	(29)	Factored CRBM (Alemi et al., 2015)	1 layer	Controlling network weights	Supervised learning and control of affect expression
	(30)	LSTM - RNN (Jain et al., 2015)	1 RNN and 2 to 3 FC layers	-	Modular RNNs organized in a graph structure
	(31)	LSTM - RNN (Crnkovic-Friis and Crnkovic-Friis, 2016)	3 layers		Kinect Data / Unsupervised dance generation
	(32)	Convolutional Autoencoders + Feed-Forward Network (Holden et al., 2016)	5 layers	Training a control network	Semi-supervised learning
	(33)	Seq2Seq with Adversarial Learning (Wang and Artières, 2017)	1 layer	Conditional inputs	Adversarial Learning
	(35)	Seq2Seq with GRU RNN (Martinez et al., 2017)	1 layer		Learning velocities using a residual architecture
	(36)	Factored CRBM (Alemi and Pasquier, 2017)	1 layer	Controlling network weights	Supervised learning and control of affect expression and navigation
	(37)	Factored CRBM (Alemi et al., 2017)	1 layer	Controlling network weights	Music-driven dance generation
	(38)	LSTM - RNN (Li et al., 2017)	3 layers	-	Alternating between ground-truth and generated movements during training
	(39)	Fully Connected (Holden et al., 2017)	3 layers	Modulating Network Weights	A phase function defines the weights of the network, so for each frame, the network weights are changed

Table 2.4: Machine Learning Methods for Movement Learning and Generation - *Continued*

Machine Learning Family		Model	Details	Factorization Technique	Remarks
Artificial Neural Networks	(40)	LSTM - RNN (Lee et al., 2018)	4 layers	Control information as extra input + loss function control	-
	(41)	GRU - RNN + Fully Connected (Pavllo et al., 2018)	2 layers	Control parameters > (FC) control network > extra input to the main RNN	-
	(42)	CNN and Autoencoders (Li et al., 2018)	3 CNN layers + 1 FC layer	-	
	(43)	Fully Connected (Starke et al., 2019)	3 FC layers	Modulating Network Weights	The weights of the network are dynamically generated at each frame based on the given control factors
	(44)	LSTM - RNN (Wang et al., 2019)	3 layers	MAP with Optimization	LSTM cells with a GAN training architecture - The output frame is modelled by a Gaussian Mixture Model
	(45)	LSTM - RNN and Autoencoders (Pettee et al., 2019)	Posture: 5 layers Dynamics: 3 layers (encoder) + 1 layer (latent layer) + 3 layers (decoder)	Latent space	One AE for learning posture and one VAE for learning dynamics
	(46)	LSTM - RNN trained in a GAN (Sun et al., 2021)	2 RNN + 5 FC layers	Extra input	-
	(47)	Fully Connected (Starke et al., 2020)	3 FC layers	Modulating Network Weights	The weights of the network are dynamically generated at each frame based on the given control factors

Table 2.4: Machine Learning Methods for Movement Learning and Generation - *Continued*

Machine Learning Family		Model	Details	Factorization Technique	Remarks
Artificial Neural Networks	(48)	Autoencoders (Ling et al., 2020)	Fully-Connected Autoencoders	Latent space + Extra input	-
Others	(3)	Linear Dynamic System (Li et al., 2002)	-	-	Motion Texture
	(18)	Multilinear Independent Component Analysis (Liu et al., 2011)	-	Optimization	-

datasets. There are, however, limitations in using some DR techniques on sequential data such as mocap as most DR models are not inherently dynamic and are not designed to directly capture the temporal dependencies of the data.

DR algorithms can be linear or non-linear. Linear algorithms include Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) among other methods. Non-linear models include t-Distributed Stochastic Neighbour Embedding (t-SNE) and Isometric feature mapping (Isomap). Among these, we briefly introduce the models used for movement generation, namely Principal Component Analysis (PCA) and Isometric feature mapping (Isomap).

Principal Component Analysis (PCA) is a linear dimensionality reduction model that uses linear combinations of the original features to derive a new, smaller set of features (Pearson, 1901; Wold et al., 1987). The new features are determined in such a way that they maintain as much of the variance in the original high-dimensional space as possible. The resulting features are referred to as Principal Components (PCs) and are linearly uncorrelated. PCs are ranked based on how much variance they account for in the original data.

Isomap is a non-linear dimensionality reduction technique that constructs a graph connecting the nearest data points (Tenenbaum et al., 2000). The graph is then used to create a lower-dimensional representation that preserves the geodesic distance[2] between all data point pairs.

The following discussion of the DR-based movement generation systems is organized based on how they model the dynamics of mocap data: 1) those that train the DR model on single frames and 2) those that train the DR model on sequences.

Dimensionality Reduction for Single Frames

This technique is based on learning a low-dimensional representation of mocap frames and then training a dynamic model such as Markov chains or Hidden Markov Models (HMM) on top of this representation to capture the dynamics. Qu et al. (2008) (12) propose a three-stage model. First, Isomap is used to learn a low-dimensional representation of the training data. Next, using linear dynamical systems, the sequences of movement in the DR space is segmented into basic units of movement, which can be assembled to create longer sequences. In the final stage, the transition matrix between the segments are calculated with the assumption that the segments satisfy a first-order Markov chain constraint.

For generating new sequences, first a sequence of segments are identified by making noise-driven transitions between the segments. Next, the low dimensional representation of each segment is mapped back to the high-dimensional pose space to produces the movement.

[2]Determined by the number of nodes on the shortest path between two nodes on the graph.

Dimensionality Reduction for Sequences

Another approach is to train the model on short segments of consecutive frames rather than individual ones. Most DR algorithms have a fixed input size. As a result, the mocap sequences with variable lengths in the training set have to be transformed into aligned, fixed length segments before fed to the learning algorithm. The typical workflow of creating such models can be summarized to the following steps:

1. *Segmentation* - The relatively long training sequences are divided into segments with variable-lengths. The segments are used to group together portions of movement that share similar patterns. The segmentation process can be automatic or manual, but it is often based on a semantic criteria. For example, the segmentation semantics can be based on the stages of a walking cycle (Tilmanne and Dutoit, 2010) (16) or more generally based on movement primitives (Min and Chai, 2012; Samadani et al., 2013) (24) (26).

2. *Alignment* - The segments are aligned to ensure all segments have the same length. Tilmanne and Dutoit (2010) (16) use the SLERP interpolation algorithm (Shoemake and Shoemake, 1985) to normalize and align the segments[3]. Samadani et al. (2013) (26) use piece-wise linear re-sampling to create fixed-length vectors. In addition, dynamic time warping is also used for sequence alignment (Min and Chai, 2012) (24) (Herrmann et al., 2017) (34).

3. *Learning* - A DR model is trained on the fixed-length, flattened segments. The resulting model maps a given input segment to a point in the DR space.

4. *Generation* - Once the low-dimensional subspace is created, one can generate a new movement segment by reversing the training procedures: 1) Choose a point in the DR space, 2) map the chosen point back into a fixed-length segment in the high-dimensional space, and 3) re-sample the generated high-dimensional segment to have the correct duration using the same re-sampling method that were used to create the fixed-length vectors.

Hierarchical Architectures

DR models are also used in hierarchical architectures. For instance, Min and Chai (2012) (24) propose an approach that creates a finite directed graph of generative movement models.

[3]To make the interpolation algorithm work on the mocap data, the Euler angles parameterizations are temporarily converted to quaternions which better supports interpolations. After the alignment, the quaternions parameterizations are converted to exponential maps, which are locally linear and more suitable for PCA.

Each node of the graph is a Functional PCA (FPCA) model and the transitions between each node is learned using a Gaussian Process (GP) model.

For movement generation without using any control parameters, a two-step procedure is followed. First, the high-level structure of the movement is generated through a random walk over the graph. Next, movement segments for each node (model) is generated by probabilistic sampling over the movement parameters. The transition movement between each node/segment are created using a blending approach introduced by Rose et al. (1998) to reduce any discontinuities around the transition points.

For control over the generation, an approach based on graph walks, probabilistic sampling, and gradient-based optimization is devised by formulating the problem as a Maximum A Posteriori (MAP) framework to find a posteriori distribution defined over three terms: transition, contact-awareness, and control at the kinematic and semantic levels. Herrmann et al. (2017) (34) further extend this work by using a k-Means tree to speed-up the optimization process.

2.5.2 Gaussian Processes

A Gaussian Process (GP) is a non-parametric, Bayesian model and can be interpreted as the generalization of the Gaussian probability distribution. Rather than modelling distributions of scalars or vectors, GP defines a distribution over the possible functions that are consistent with the training data. GP is commonly used for classification and regression problems, as well as data visualization (Rasmussen and Williams, 2005). Another application of GP is creating latent variable models, in which the GP learns a non-linear mapping from a latent, low-dimensional manifold to the observations.

Modelling the Dynamics

Vanilla GP is not an inherently dynamical model. As a result, two approaches are used to model movement dynamics: combining GP with a dynamical model, and extending GP to explicitly model the dynamics.

In the method proposed by Taubert et al. (2012) (22), movement dynamics are modelled using a Hidden Markov Model (HMM) while the mocap frames are modelled by a Gaussian process latent variable model (GPLVM). The model generates hand-shake movements for a chosen category (neutral, fearful, happy, angry, and sad). As shown in Fig. 2.5, the resulting model consists of three layers: the bottom layer is the GPLVM-single, in which the movements of one individual actor are mapped onto a 3-dimensional latent variable while capturing the variations with respect to parameters such as actors, trials, emotional category, and time. The interaction layer (GPLVM-interaction) learns a 3-dimensional latent variable from a 6-dimensional observation variable that is created by the learned bottom-layer model for each pair of interacting actors. In the top layer (HMM-dynamic), a left-to-right HMMs

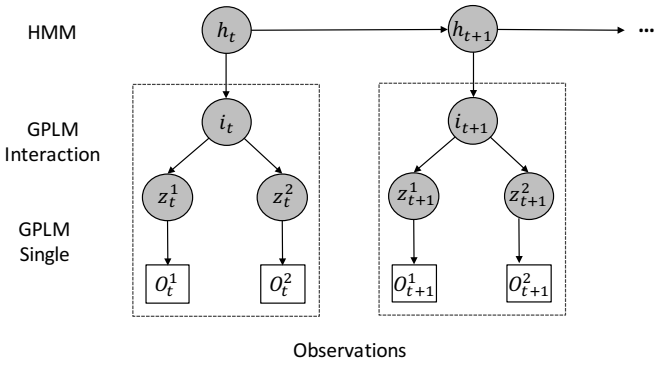


Figure 2.5: The three-layer model described in (Taubert et al., 2012). In the static model, shown inside the box, at each frame t, the handshake data of two different actors (O^1 and O^2) are mapped into two separate latent spaces (z^1 and z^2). The latent representation of both actors are then combined and mapped into another latent space i, which represents the interactions between the two actor. In the dynamic model, a hidden Markov model with hidden unit h is trained on the representation in i to learn the dynamics of movement.

with seven states learns the temporal evolution of the latent variable in the interaction layer. For each emotion category, a different HMM is learned.

Wang et al. (2008) (10) extend the GPLVM with a dynamical model over its latent space to learn the temporal structures of sequences. The resulting model provides generative mappings between the input observations and a low-dimensional and sequential latent space. GPDM can model first-order, as well as higher-order Markov chains, which can be used to learn the speed and acceleration of movement as well.

Control

In order to control the generated movement, Wang et al. (2007) (9) introduce Multifactor Gaussian Process Model (MF-GPM) to learn and generate cyclic locomotion. The MF-GPM includes a low-dimensional latent space of multiple movement factors, as well as a mapping from the latent space to the high-dimensional observations. The MF-GPM is capable of learning a generalized model, which allows it to generate movements with factor combinations that do not exist in the training data.

Another approach to control the movement is to map the points in the latent space to movement features. Levine et al. (2012) (25) use the low-dimensional embeddings learned by a GPLVM to create an interactive and controllable movement generation model. Each point in the low-dimensional latent space corresponds to a Gaussian distribution of poses. To be able to generate a continuous movement from a curve, similar points in the latent space have to be densely positioned. To increase the connectivity of the points in the latent space, Levine et al. (2012) (25) extend the standard GPLVM with a connectivity prior to

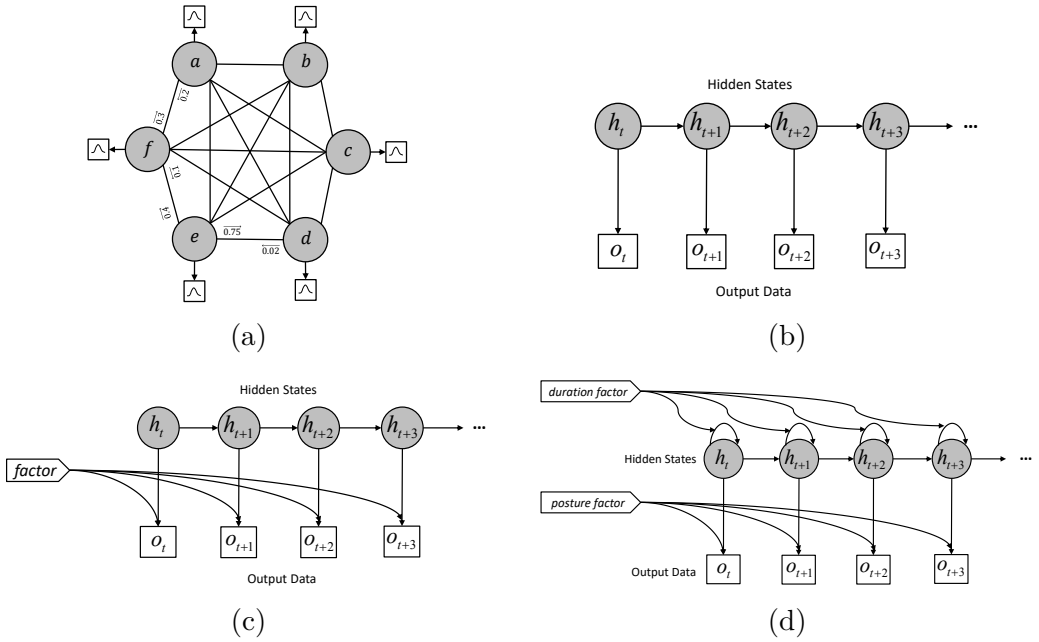


Figure 2.6: HMM Architectures: (a) A simple HMM with 6 states. The numbers on the edges denote the probability of a transition from the source state to the target state (not all probabilities are shown.) (b) An HMM unrolled through time with the sequence of the hidden states on the top and the corresponding observations on the bottom. (c) An HMM with its observation (output) probability distributions conditioned on a factor variable. (d) A Hidden Semi-Markov Model (HSMM) with two factor variables, one for the duration of each state and one for the outputs.

learn a low-dimensional space of movement that can be used to generate transitions that do not exist in the training data.

2.5.3 Hidden Markov Models

Hidden Markov Model (HMM) is a statistical model that learns the underlying stochastic process from sequential data in the form of a sequence of hidden states (Rabiner, 1989). The underlying process is not directly observable, but can only be inferred through the outputs of the states which are observable (visible).

An HMM consists of a finite set of hidden states. At each time frame, the model makes a transition to a new state with a probability that is determined by a transition matrix (Fig. 2.6a). When applying HMMs on the data, we make the assumption that in order to predict the future state of a system (e.g., the next mocap frame), we only need the current state of the system. A hidden state of the HMM therefore does not store the information about the past history of the sequence, but only the information needed to model the current observations.

In its simplest form (Fig. 2.6b), to train an HMM one has to determine the probabilities that each hidden state can be the initial state of the system, the state-to-state transition probability matrix, and the probability distributions that model the output of each state. Classical HMMs are trained using variations of the Expectation-Maximization (EM) algorithm, such as the Baum-Walch algorithm (Rabiner, 1989). Generating new movements using an HMM involves creating a sequence of hidden states, and then generating the pose for each state by sampling from its probability distribution. Creating the hidden state sequence can be done manually, e.g., if they are interpretable such as a particular walking phase, or can be done automatically using the Viterbi algorithm (Rabiner, 1989). Hidden Semi-Markov Model (HSMM) extends classic HMM by modelling the dwell-time of each of its hidden states. This ability makes HSMMs suitable for modelling the variations in the duration of movement (Fig. 2.6d).

In using HMMs to generate movement, one has to address a number of problems. Mainly, HMMs have a limited expressive power. Typically, each hidden state of an HMM models a single frame of data through its output probability distribution. This makes HMMs inefficient to model datasets that have a diverse repertoire of movements or just different variations of the same movement (e.g., different walking styles as shown in the literature). The inefficiency in modelling movement variations also makes it difficult to control the generated movements as it is often the case that the control is around changing the variations of the same functional movement. In the following, we review the approaches that are used to address the aforementioned limitations.

Parameterizing the Probability Distributions

One of the approaches to allow HMMs to learn the variations around an average model, and thus to improve their expressive capacity, is to define the mean and the covariance matrix of the probability distribution of the state observations as functions of the factors of the variation in the data (Fig. 2.6c). In this way, each state learns a space of probability distributions for its possible observations. By changing the values of the factors during both the learning process and inference time, one modulates the behaviour of the distribution and thus can control the characteristics of movements being generated. We should note that this approach can only be effective if the distributions of the various movement factors in the training data can be described around an average distribution.

Examples of this approach include Parametric Hidden Markov Model (PHMM) (Herzog et al., 2008) (13) that uses Gaussian distributions with their means and covariance matrices being a function of the factor variables, HMM/Mix-SDTG which uses Mixtures of Stylized Decomposable Triangulated Graph (Mix-SDTG) for the distribution of the observations, and the Stylistic Hidden Markov Model (SHMM) used in *Style Machines* (Brand and Hertzmann, 2000) (1) is an HMM with its parameters (e.g., the means and covariances of the observation probability distributions) being functionally dependent on a factor variable.

Generic and Specific Models

In this group of approaches, instead of directly training an HMM for a desired variation, a new HMM is created with its model parameters derived by manipulating (e.g., interpolating or transforming) the parameters of the probability distributions of the state observations of one or more pre-trained HMMs.

For example, the Style Machines model (Brand and Hertzmann, 2000) (1) simultaneously learns a generic HMM as well as a group of style-specific HMMs using an entropy minimization algorithm in an unsupervised manner. The generic model captures the movement mechanisms that are shared among all the styles. Each style-specific model then only captures a variation of the generic movement.

Herzog et al. (2008) (13) train a group of individual classic HMMs, one for each factor value in the training data. This results in a group of HMMs each representing an average model for a particular factor value. To generate movements based on desired factor values between the sample models, the authors use component-wise linear interpolation of the means and covariance matrices of the observation distributions to derive a new HMM that for that particular factor value that generates the desired movements. In order for the interpolation to work, the states of all of the sample HMMs should be synchronized, meaning that for each state in one HMM, there is an equivalent state in all other HMMs. As a result, the movements in the training data are aligned so that all the individual HMMs have the same number of states that point to the same parts of movement.

In a different approach, instead of learning style-specific HMMs, Tilmanne et al. (2012) (23) only train a generic (average) walking model from a relatively large number of training sequences. The style-specific HSMMs are then created by transforming the parameters of the generic model to produce a particular walking style using linear transformations that are borrowed from speech synthesis applications (Gales, 1998; Yamagishi et al., 2009). This transformation only needs a relatively small amount of sample data for each specific style, making it easier to derive models for multiple styles without requiring a large collection of style-specific training data.

In a later work, Tilmanne et al. (2014) (27) extend the previous approach. For the generation, the HSMMs are 'unwrapped', i.e. the transition matrix is replaced with an explicit model. Next, to ensure the smooth trajectory of the output, the Maximum Likelihood Parameter Generation (MLPG) algorithm (Tokuda et al., 2000) is used over the strict Maximum Likelihood criterion. This model allows for choosing the factors of the generated movements (styles), as well as interpolating models based on a weighted sum of the model parameters in order to blend, inhibit, exaggerate, or inverse the movements factors.

Sequence Segmentation and Model Concatenation

To train HMMs on longer sequences more effectively, some works follow a divide and conquer approach and divide the longer sequences into shorter segments, group similar segments together, and a train separate HMMs on each group. For generation, first the desired arrangement of the HMMs, corresponding to the desired arrangement of shorter movements, is defined. Next, a segment is sampled from each HMM. Finally, the segments are concatenated to generate the full sequence.

Semantically, this segmentation can be interpreted along the notion of movement primitives. For example, Yamazaki et al. (2005) ④ manually segment walking cycles into four primitives: the L-step, which is the back-to-front movement of the left leg, the R-step, which is the back-to-front movement of the right leg, and two primitives for the beginning and the end of a walking cycle. Tilmanne et al. (2012) ㉓ follow a similar approach, but use five primitives for the walking cycle. In both cases, a separate HMM is trained for each primitive.

Hierarchical Architectures

Another group of works also tackle the complexity of learning a diverse repertoire of movement by following a divide and conquer approach, but they assemble the models in hierarchical architectures instead of linearly concatenating individual models (Tanco and Hilton, 2000; Wang et al., 2005; Kulić et al., 2011) ② ⑤ ⑲. A hierarchical architecture gives such models the advantage over single HMMs or concatenation approaches. First, the ensemble model has an increased capacity. Second, while classical HMM is based on a first-order Markov model, a hierarchical architecture allows for capturing longer temporal dependencies within the data. Third, hierarchical architectures provide more flexibility in arranging the movements.

A typical workflow of this group involves segmenting movements, clustering similar segments, training a model for each cluster, and creating a hierarchical organization. In the following, we discuss each of these stages as well as the generation procedure.

Segmentation - Similar to the sequence segmentation and model concatenation approach, the segmentation can be done manually by a person or algorithmically. Likewise, the segmentation criteria is loosely based on the notion of movement primitive. Kulić et al. (2011) ⑲ use a probabilistic segmentation technique that automatically finds segments of the movement that have similar probability distributions. (Wang et al., 2005) ⑤ use heuristics from biomechanics to automatically segment the movement into movement primitives.

Clustering - The clustering can be down offline during the training process or in an online manner. To identify the clusters during the training process, algorithms such as the *K-means* clustering algorithm (Tanco and Hilton, 2000) ② and the Expectation-Maximization clustering algorithm are used (Wang et al., 2005) ⑤. To incrementally cluster the segments

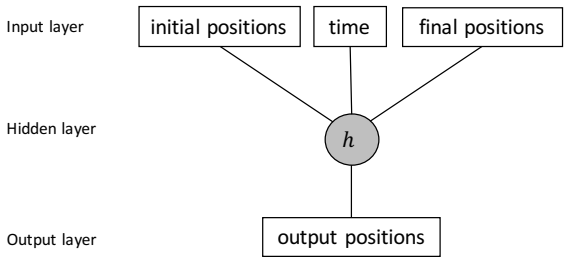


Figure 2.7: The architecture of the perceptron-based neural network using by Lin et al. (2008).

that are streamed in an online manner, Kulić et al. (2011) (19) first model each incoming segment using an HMM. This HMM is then compared to the other HMMs in a tree structure and a new cluster (tree node) is generated if the difference is significant. Otherwise the segment is added to the most similar cluster.

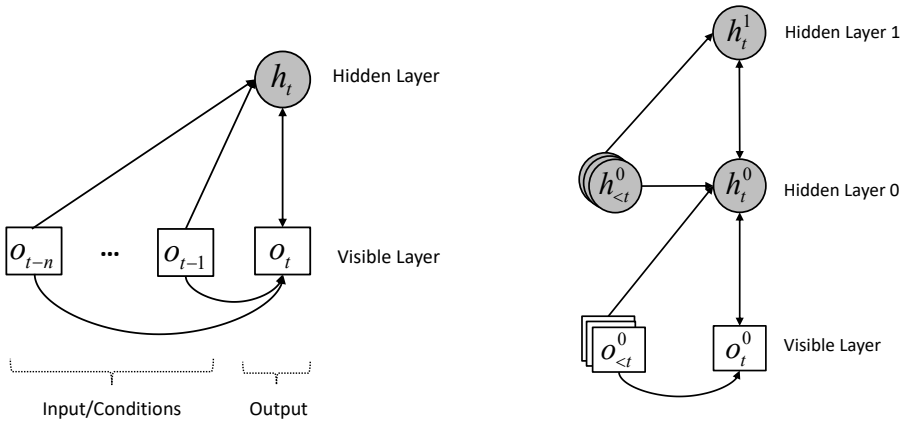
Model Learning - For each cluster, a separate HMM is trained. The HMM learns the average movement, representative of the movements in the same cluster.

Hierarchical Organization - Graph or tree structures are used to organize the models. The nodes typically correspond to the clusters. Each node is represented by a model responsible for generating movement samples from their clusters. The graph or tree structure then represents the relationships between the nodes (Tanco and Hilton, 2000) (2) (Wang et al., 2005) (5). It is also possible the change the structure as more data becomes available. Kulić et al. (2011) (19) take advantage of the flexibility of hierarchical organizations and propose a method for learning movements from observations during an on-line, continuous process. New movement segments are incrementally organized into a hierarchical tree structure in which each node represents a cluster. If a new movement primitive is introduced to the model at any stage, the organization of the tree is modified by adding a new node.

Generation - For generating new movements, first a sequence of key-frames, represented by their corresponding nodes, are determined by a random or a constraint-based walk over the structure. Next, for each node a movement segment is generated using the HMM that models the cluster. Finally, the segments are concatenated according to the order of the nodes. Further optimizations can be performed to ensure that the movement is continuous in adjacent segments (Wang et al., 2005) (5).

2.5.4 Artificial Neural Networks

In the following, we review studies that use artificial neural networks to learn and generate movement. We break down our review based on the type of the neural network used.



(a) (b)

Figure 2.8: (a) The architecture of a single-layer Conditional Restricted Boltzmann Machines (CRBM) with n previous time steps as the conditional inputs. (b) A Conditional Deep Belief Network (CDBN) built from two CRBMs. The $o_{<t}$ and $h_{<t}$ represent the data history vector $t-1, ..., t-n$ where n is the number of past time steps that are connected to the units at the current time step.

Perceptrons

Lin et al. (2008) (11) use feed-forward neural networks in combination with optimization techniques for learning and generating the trajectory of a humanoid arm lifting objects. The perceptron-based neural network, as shown in Fig. 2.7, works as a function approximator for the joint positions. The network has one hidden layer with ten hidden units. Its input layer consists of the frame number (time), initial joint positions, final joint positions, and the total number of frames. The output layer predicts the position of the joints at a particular time during the lifting movement. The output is then applied to an optimization model to ensure that the initial and final positions match the desired values. The network is trained using the back-propagation algorithm.

Self-Organizing Mixture Networks

Self-Organizing Mixture Network or SOMN (Yin and Allinson, 2001) is a type of artificial neural network that estimates mixture distributions using a self-organizing, unsupervised approach. Wang et al. (2006a) (6) use a SOMN of parametric Gaussians and introduce an approach called *key-styling* to generate movement animations. The model learns a probabilistic mapping from the low-dimensional space of factors to the high-dimensional pose space, while the mapping is controlled by a style (factor) variable. Unlike the original SOMN, this model uses a conditional probability distribution to learn the effects of different values

of movement factors. The value of each dimension of the factor variable is determined in a supervised manner via the annotations of the training data.

Restricted Boltzmann Machines (RBM)

RBMs are generative neural networks consisting of a visible layer and a hidden layer in such a way that the visible units are fully connected to the hidden units, but there are no visible-to-visible or hidden-to-hidden connections (Smolensky, 1986). RBM is an energy-based neural network in which an energy value is associated to each configuration of its visible and hidden units. Training the model is then analogous to finding the network state with the lowest energy. Once trained, the RBM models the distribution of the data observed by their visible units. RBMs are trained by the Contrastive Divergence (CD) algorithm (Hinton, 2002). It is also possible to stack RBMs to create Deep Belief Networks (DBN) (Hinton et al., 2006) and achieve more representational power than single-layer models.

In the following, we discuss how the RBM-based models address issues such as modelling the dynamics of the movement, training deeper models, and controlling the generation.

Modelling the Dynamics - RBMs model a single frame of data. Conditional RBMs (CRBMs) extend RBMs by conditioning the model on a fixed number of additional inputs, which can be used to condition the probability of the next frame on a fixed sequence of past frames (Taylor et al., 2006) (8). This is done by adding two sets of new connections to the model: Autoregressive connections from each extra input (past frames) to the current visible units, and the connections from the extra inputs to the hidden units (Fig. 2.8). The former connections model the linear, temporally local structures, while the latter connections model the non-linear and higher-level structures (Taylor et al., 2006) (8). CRBMs can be trained similar to RBMs using the CD algorithm.

Control and Generation - A CRBM does not support an explicit representation of movement factors but a single model can learn and generate different movement variations (e.g., walking and running) if they exist in the training data. In this setting, the type of the movement to be generated is specified by seeding the model (a small number of frames used as the first set of conditional inputs). For example, if frames from a walking movement are used to initialize the model, it will generate walking movements and if frames from running are used, it will generate running movements, with occasional uncontrolled transitions between the two (Taylor et al., 2006) (8).

An updated version of CRBM (Taylor, 2009) uses soft-max labels as extra input conditions to control the factors of movement during the generation. However, the authors determined this technique to be inefficient as each hidden unit also receives many connections from the past and the current visible units, which diminishes the influence of the soft-max labels.

Further research in explicitly controlling movement factors using CRBM resulted in Factored CRBM (FCRBM) (Taylor and Hinton, 2009) (14), a model which supports more rep-

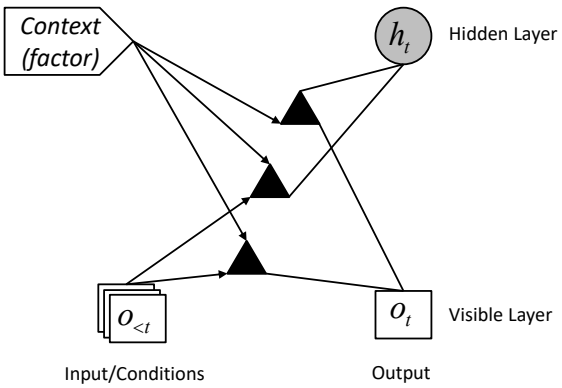


Figure 2.9: The architecture of a Factored CRBM with interactions gated by real-valued factors.

resentational capabilities and effective control over the generation. As depicted in Fig. 2.9, FCRBM uses three-way connections that allow a third unit (the context unit) to control the interactions between the visible and hidden units. Thus, the user can control the factors of the movements being generated. The context unit of the FCRBM can be binary or real-valued, allowing it to capture and represent different types of movement factors from categorical to continuous factors. The interaction of the factors with the model is learned in a supervised manner using annotated data.

FCRBM is used for controlling the generated movement through factors such as the walking speed and stride length (Taylor and Hinton, 2009) (14), the affective state communicated by the mover (Alemi et al., 2015) (29), as well as the walking direction and the personal movement signature of the mover (Alemi and Pasquier, 2017) (36). Note that as the context unit can be multi-dimensional, multiple factors can be used in a single model, as in WalkNet which controls the affective state, the walking direction, and the personal movement signature of the mover at the same time (Alemi and Pasquier, 2017) (36). Furthermore, one can interpolate or extrapolate the factors to create new characteristics that did not exist in the training data. For example, the model by Alemi et al. (2015) (29) generates a full spectrum of affective states and transitions by only training the model on nine discrete affective states.

In addition to the aforementioned factors, FCRBM is shown to be able to model complex multi-modal relationships such as audio and movement. Chiu and Marsella (2011b) (20) use a an FCRBM in a hierarchical architecture (Fig. 2.10.b) to generate gestures using the prosody of speech as the controlling factor. Using a set of training data that includes motion capture recordings of gestures accompanied by the voice recordings of the actors, the model learns the relationship between the prosody of speech and the movement. Alemi et al. (2017) (37) propose GrooveNet, a music-driven dance generation model using FCRBM. GrooveNet uses

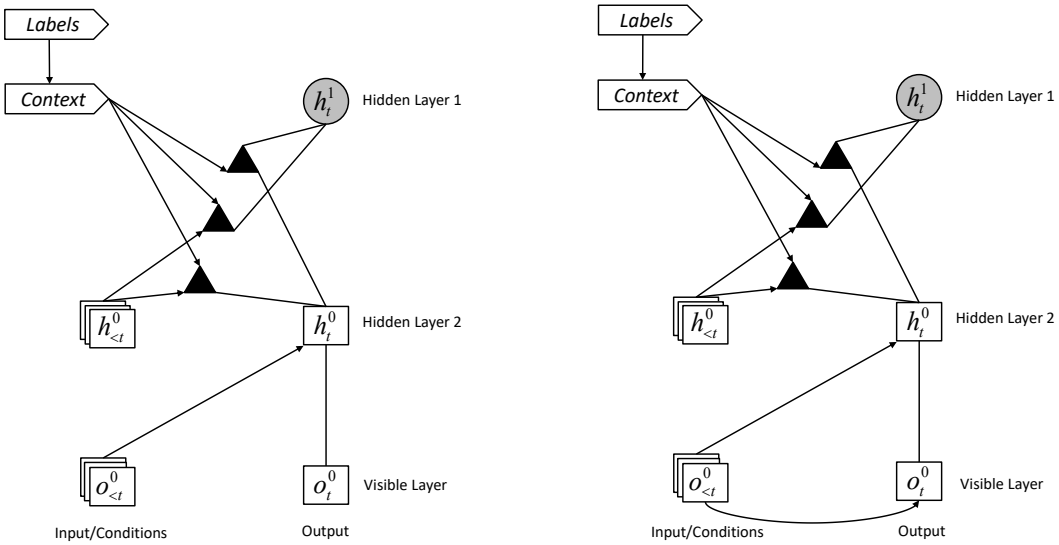


Figure 2.10: (a) The architecture of a Hierarchical FCRBM (HFCRBM) with a Reduced CRBM as the first layer and an FCRBM as the top layer. (b) A modified HFCRBM with a CRBM as the first layer and an FCRBM as the top layer.

an 84-dimensional vector of audio features for the context units, allowing it to control the movement generation based on the audio features.

In a different way of controlling the generation, Chiu and Marsella (2011a) (21) propose an approach to blend two movement with different characteristics. Instead of directly interpolating the joint rotations, the interpolation is done on a space of latent representations learned by the model. To interpolate between two factor values, we first generate a sample for each value from the FCRBM. The generated samples are effectively two representations of the hidden layer of a CRBM. We then create a weighted sum of these two representations, and generate the corresponding new sample from the CRBM.

Adding Depth - Chiu and Marsella (2011a) (21) propose an extension of the CRBM called the Hierarchical FCRBM (HFCRBM). An HFCRBM consists of a Reduced CRBM as its bottom layer and an FCRBM as the top layer (Fig. 2.10a). A Reduced CRBM is the same as a CRBM except that it does not include the autoregressive connections. In an HFCRBM, the input visible data are fed into the Reduced CRBM. Once the Reduced CRBM is trained, an FCRBM is trained on the features discovered by the hidden layer of the Reduced CRBM as its input. This way of stacking the models together without the autoregressive connections ensures that during the generation, the visible data are only affected by the top layer and its labels and not the past visible data. A modified version of the HFCRBM (Fig. 2.10b) uses a CRBM with its autoregressive connections instead of the Reduced CRBM.

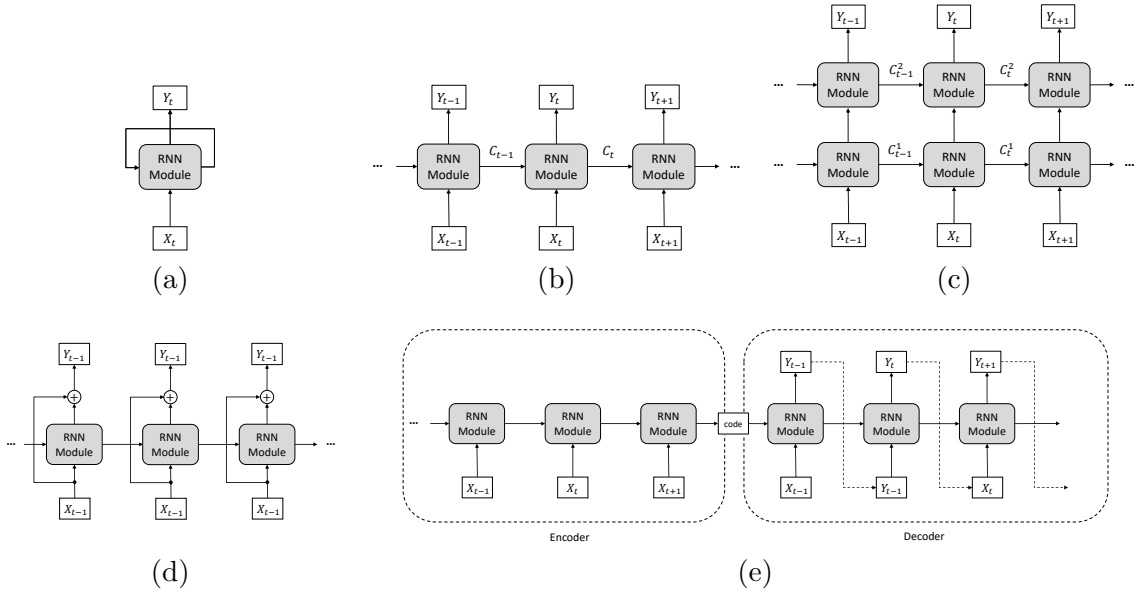


Figure 2.11: RNN Architectures. (a) A simple RNN. (b) The same RNN unrolled through time. (c) a two-layer RNN architecture. (d) RNN with residual connections from the input to the output. (e) A sequence-to-sequence architecture.

Recurrent Neural Networks (RNN)

RNNs are machine learning models that process the input data sequentially by using the same neural network module on every element of a sequence. The module takes the input at each time steps, as well as its own hidden state from the previous time step and calculates an output vector while updating its hidden state (Fig. 2.11a). As a result, the output and the hidden state of the network at each step depend on the hidden state of the network at the previous step. This chain of dependencies (Fig. 2.11b) going back to the first step allows the hidden state of the network to maintain information about the sequence across all time steps and act as a (lossy) memory cell. As a result, RNNs are able to capture long-term dependencies in the data and model sequences with arbitrary lengths. RNNs are powerful tools to model sequential data as demonstrated by the research across multiple domains such as natural language processing, speech, computer vision, and music.

In the following, we present and discuss different approaches to use RNNs to learn and generate movement, and how they address common problems in using RNNs such as the vanishing or exploding gradients, fading of the model output when modelling real-valued data, error propagation, training, and network architecture and size. We also look at the problems more specific to movement generation such as the control and generation methods, loss functions, and input and output representation.

Vanishing or Exploding Gradients - One of the challenges in using gradient-descent-based learning algorithms to train neural networks with long-term dependencies is that during the backpropagation, after a few steps the gradient either vanishes or explodes.

A solution to the vanishing gradient problem is to use gated RNN cells instead of simple ones. Gates are neural networks that are designed to control what parts of information should pass through them. They let the network determine what information to remember and what information to forget, which in turn helps the network to better learn the long-term dependencies in sequences. There are two gated cell types are commonly used in the literature, namely the Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Graves, 2013) and the gated recurrent units (GRU) (Cho et al., 2014) which simplifies the LSTM design and has fewer parameters.

To overcome the exploding gradients, a simple practice is to clip the gradients before updating the network parameters during the training to prevent them from getting too large. For a more detailed discussion of the vanishing and exploding gradient problems, refer to the book by Goodfellow et al. (2016).

Dying-Out Outputs - A problem that occurs when training RNNs with real-valued data is that the model is prone to converge to predicting an average output after a few iterations. This is often referred to as the *stagnating output* or the *dying out output* problem in the literature. Bishop (1994) show that this occurs due to using the mean squared error (MSE) loss function on the real-valued data, pushing the model to output a vector close to the mean vector.

A common solution to this problem is to train the model to predict the probability density function for each dimension of the data instead of directly predicting the values (Schuster, 2000). The values of each dimension can then be sampled from the predicted distribution. Mixture Density Networks (MDN) (Crnkovic-Friis and Crnkovic-Friis, 2016) (31) (Pettee et al., 2019) (31) and Gaussian Mixture Models (GMM) (Fragkiadaki et al., 2015) (28) (Wang et al., 2019) (44) are two common approaches used to learn the probability density of the data (Bishop, 1994).

Generation - New movements are sampled from RNNs in an iterative manner where the network is in a closed loop. At each iteration, the output of the model from the previous iterations, as well as the hidden state of the network are used to generate the next frame.

Control - Controlling the generation with RNNs using high-level factors is challenging as it requires the generation of a sequence being conditioned on another sequence with a different temporal granularity (the control factors could change at a different rate as the frame rate of the movement).

A common approach to control the output of RNN models is to use the control descriptors as extra input to the network by concatenating the descriptors with the movement features. Tang et al. (2018) add the sequence of the audio features to the input data to generate dance movements. Lee et al. (2018) (40) introduce a model for interactively control

the movement generation using a multi-objective set of control parameters. The animated character controlled by this model will have a repertoire of actions to choose from (the functional dimension of the movement). Each action is further parameterized using one or more factors specific to that action (the planning dimension). Pavllo et al. (2018) (41) pass the control parameters through a small network with two fully-connected layers before feeding them to the RNN as extra input.

Error Propagation/Accumulation - As mentioned before, new movements are generated in a closed loop. In this setting, a small divergence in a generated frame can be propagated to the future frames and eventually cause the model to generate either diverging values or converge to a fixed output. This can be explained by the difference between how recurrent networks are typically trained and how they are used for inference. During the training, the input to the model is always from the ground truth data. However, when used for inference, the model is fed with its own predictions from the previous time-steps to predict longer sequences. As a result, the model only learns to predict outputs from prevents the model to learn to recover from its own mistakes effectively (Bengio et al., 2015).

This problem is addressed in a number of different ways in the literature. Li et al. (2017) (38) train the network on sequences that periodically alternate between using the ground truth data from the training set and the past generated outputs of the RNN (Fig. 2.12). Pavllo et al. (2018) (41) use a curriculum schedule (Bengio et al., 2015) to train the model. The training algorithm is similar to the alternating approach, except that it chooses ground-truth data over the model's own predictions as the input with a probability of p. The training starts with $p = 1$ but p is gradually decreased by an exponential factor at each time step, increasing the probability of choosing the model's previous predictions as its input. The decay factor is treated as a hyper-parameter and an optimal rate has to be determined through experimentation. Martinez et al. (2017) (35) use a simpler learning approach in which they only feed the predictions of the network as the input to the decoder RNN instead of the ground-truth data during the training, therefore eliminating the need for finding the optimal decay factor.

Loss Function - RNNs are trained by the gradient descent algorithm and therefore require a loss function to measure the prediction error during the training. The majority of models use the mean squared error (MSE) between the predicted frame and the ground truth frame as the loss function. There are, however, a few works that use a different loss function.

Extra terms can be added to direct the model to follow certain constraints. In the work by Lee et al. (2018) (40), the loss function consists of three terms: movement loss, contact loss, and rigid body loss. The movement loss measure how different the predicted joint positions are from the ground-truth. The contract loss is added to minimize the artifacts when interacting with external objects, such as foot sliding on the floor or touching the ball at the correct time and location. The contact loss is calculated from the annotated binary

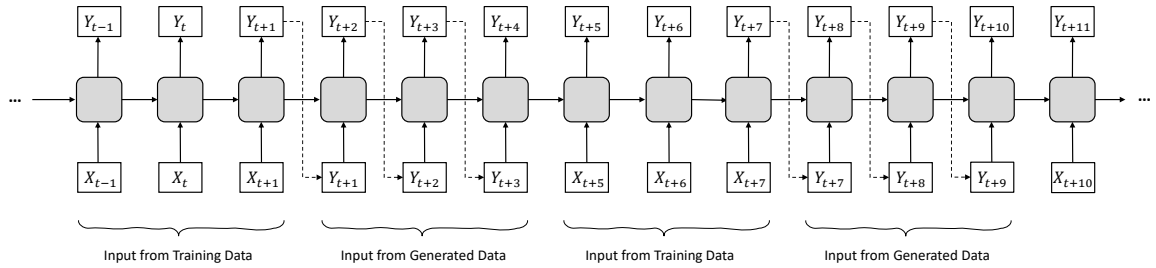


Figure 2.12: Training an RNN by alternating between training and generated data.

contact information. The rigid body loss ensures that the body limbs maintain the correct length during the generation.

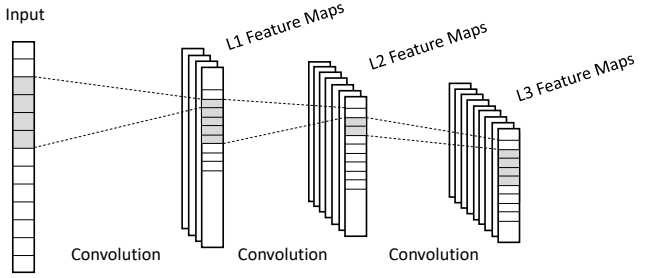
Pavllo et al. (2018) (41) argue that while training the model on rotations has the advantage of better interoperability with animation pipelines, defining the loss function over joint positions is more suitable for training the network. Defining the loss function over the rotations causes small errors in the rotation of the joints higher in the skeleton hierarchy to propagate to the rotations of the joints down the skeleton, such as the end-effectors. Furthermore, because of their periodic nature, rotation representations have discontinuities which can result in incorrect error calculations (e.g., $|2\pi - 0| = 2\pi$ while both angles are the same.). As a result, Pavllo et al. (2018) (41) train the model on joint rotations represented in quaternions, but use a differentiable forward kinematics function to calculate joint positions in order to define the loss function over the position values.

Sun et al. (2021) (46) propose a loss function based on motion consistency and kinematic factors. It consists of a weighted sum of three terms: the Euclidean distance between the feature vectors represented in exponential maps, the difference in joint positions, root direction, and root orientation, and the difference in the speed of the joints as well as the angular speed of the root orientation.

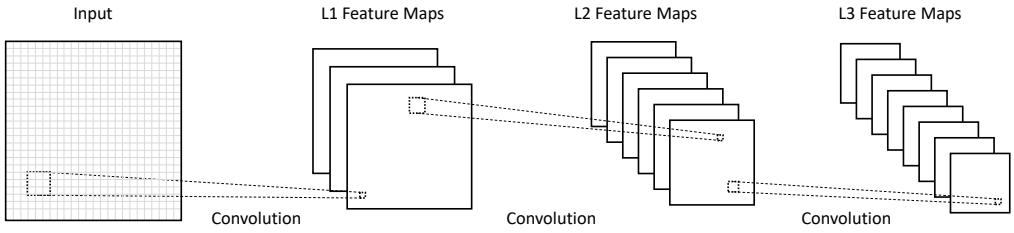
Network Architecture - The architectures of recurrent networks vary from stacked RNNs to encoder-decoder networks, sequence to sequence architectures, residual connections, and graph structures.

It is common to use fully-connected networks before the input of the RNN and after the output of the RNN. These networks are used to learn representations of the data or the control parameters or to add non-linearity to them (Fragkiadaki et al., 2015; Lee et al., 2018; Pavllo et al., 2018; Sun et al., 2021) (28) (40) (41) (46).

As mentioned before, some networks predict the parameters of the probability density functions of the output instead of directly predicting the values. In such cases, a separate fully-connected network is used for each parameter (such as mean and standard deviation) of the distribution (Crnkovic-Friis and Crnkovic-Friis, 2016) (31) (Fragkiadaki et al., 2015) (28) (Wang et al., 2019) (44). The activation functions of these layers are often chosen to ensure that the predicted values are within the valid range.



(a)



(b)

Figure 2.13: Convolutional Neural Networks (CNN). (a) convolutions over a 1-dimensional input (e.g., mocap sequence). (b) convolutions over a 2-dimensional input (e.g., a static image).

Martinez et al. (2017) (35) use a sequence-to-sequence architecture (Fig. 2.11e) that consists of an encoder GRU-RNN and a decoder GRU-RNN that share the same set of weights.

Another architectural pattern that is used for movement generation is to use residual connections at the output (Martinez et al., 2017) (35) (Pavllo et al., 2018) (41). These connections add the predicted values to the joint angles from the previous frame (Fig. 2.11d). The residual connections implicitly push the network to internally model the velocities of the joint rotations instead of their absolute values, which can help reduces the amount of drifting in the predictions.

In a different approach, Jain et al. (2015) (30) use multiple RNNs in a modular graph to model spatio-temporal data. In the proposed architecture, called the Structural-RNN (S-RNN), the nodes of the graph represent the modules of the problem and the edges represent the spatio-temporal relationships between the modules. In an S-RNN, RNNs model the temporal edges, the spatio-temporal edges, as well as the nodes. Parameter sharing is applied to RNNs to train the models more efficiently.

Convolutional Neural Networks (CNN)

CNNs are neural networks designed to process the data following a topological structure. Each convolutional layer has a number of filters, each with the same structure as the input data but much smaller in size. Each filter is applied to multiple regions of the input to cover the whole input space. This results in the filters being able to learn features that are invariant to the location within the input space (Fig. 2.13). Besides the convolutional layers, a CNN typically consists of pooling (subsampling) layers, as well as fully-connected layers at the output. CNNs are trained using the backpropagation algorithm similar to the fully-connected networks.

While CNNs are popular for computer vision applications in which they learn the spatial features in the images organized in 2D structures (Fig. 2.13b), they can also perform convolutions over the time dimension of the data to model the temporal features (Fig. 2.13a). As a result, CNNs are also used to model the temporal (e.g., 1D time series) and the spatio-temporal data (e.g., video). Modelling time series by CNNs work by breaking the input sequences to fixed-length segments and applying convolutions over the time dimension. Consequently, the filters learn features that are invariant to their location within the time dimension.

Architecture - Network architectures vary from feed-forward networks (Holden et al., 2016) (32) to encoder-decoder architectures. Holden et al. (2015) use a convolutional autoencoders to learn human movement manifolds that can be used by generative models. Li et al. (2018) (42) devise an encoder-decoder architecture to capture both the short-term (e.g., joint motion) and long-term temporal (e.g., the functional factors) characteristics of movement by training two encoders with a similar structure but at two different temporal scales. The long-term encoder module learns to map long sequences to their corresponding long-term hidden variables. Similarly, the short-term encoder module learns the short-term dependencies between the adjacent frames, mapping each input to a hidden variable representing the short-term features. The decoder module takes the concatenated long and short-term hidden variables and predicts the next segment of the movement.

Control and Generation - As opposed to the iterative, frame-by-frame generation approaches used by RBM or RNN based models, CNNs generate a movement segment at once. To control the generation, (Holden et al., 2016) (32) train a separate CNN to control the movement factors. This network creates a regression model that maps the high-level parameters (factors) to the movement output. While the main generative network is trained over the whole database, the control network is trained over a subset of movements that is desired to be controlled. Furthermore, since the control network is trained independently from the generative network, it allows for learning different control networks for different movement types using the same model.

Autoencoders

Autoencoders (AEs) are neural networks designed to learn a representation of the data in an unsupervised manner. An autoencoder typically consists of an encoder module, which maps the input to a latent (hidden) representation, and a decoder module, which attempts to reconstruct the input from the latent representation. The latent representation often has a lower number of dimensions compared to the original data, which causes the network to learn the most useful features of the data while ignoring the noise.

Architecture - The encoder and decoder modules themselves can be built using a variety of neural networks such as fully-connected layers as well as RNNs and CNNs.

Pettee et al. (2019) (45) use two different AEs, one for learning a representation of single poses and another one for learning a representation of sequences of poses based on the pose representation. The encoder and decoder modules of the pose AE each have two layers of fully-connected networks with 64 units, separated by a middle layer with 32 units which in turn learns a 32-dimensional representation of poses. The sequence AE is trained on this pose representation and is built based as a variational AE (VAE) (Kingma and Welling, 2014). In a VAE, rather than learning a fixed representation, the encoder learns the probability distribution that describes the latent space. Pettee et al. (2019) (45) use a three-layer RNN with 384 LSTM cells for each of the encoder and decoder modules and a fully-connected network with 256 units to represent the probability distribution parameters (mean and standard deviation).

Ling et al. (2020) (48) propose a movement generation framework trained based on autoregressive conditional VAEs. Instead of using RNNs or CNNs, Ling et al. (2020) (48) use fully-connected networks for both the encoder and the decoder components of the VAE. The encoder, a three-layer network, maps the previous and the current frames to a latent representation. The decoder module takes the representation of the latent space, as well as the previous mocap frame, and predicts the next mocap frame. Instead of using a single decoder, six decoders are trained in a Mixture of Expert (MoE) architecture that uses a gating network that predicts the blending coefficient of the six decoders.

Control and Generation - Generating new samples using AE is done by first choosing a point in the latent space, and using the decoder module to map the point to the original mocap space. In the two-stage encoding of the work by Pettee et al. (2019) (45), generation starts by choosing a single point in the sequence latent space and use its decoder to get a sequence of points in the pose latent space. Using the pose decoder, the pose representation of each frame is transform to the mocap representation.

Ling et al. (2020) (48) demonstrate controlling the generated movements within a latent space, learned via a VAE. In this approach, one can control the movement using random-walks and sampling-based control methods, as well as training a control policy network using reinforcement learning.

Generative Adversarial Networks (GANs)

GANs provide a framework for training generative models using an adversarial setting where two networks are trained simultaneously: a generator network, tasked with generating a new sample that belongs to the input distribution, and a discriminator network, which is tasked with detecting if a given sample belongs to the input distribution or not (is it "real" or "fake") (Goodfellow et al., 2014). The goal of the generator network is to trick the discriminator, while the goal of the discriminator is to get better at detecting fake inputs.

Wang et al. (2019) (44) use the generator network of GAN to refine the outputs of the RNN-based movement model. While the RNN model is trained on a large dataset of mocap data, the GAN is trained on a smaller subset of mocap data, which allows fine-tuning the outputs for the subset of the movements.

The generator module of the GAN consists of a fully-connected network followed by an LSTM layer and another fully-connected network. The GAN generator refines the movement features such as the positions and velocities of the end effectors. The GAN discriminator uses a bidirectional LSTM with fully-connected layers attached to its input and output.

Sun et al. (2021) (46) use a conditional GAN framework to train a music-driven dance generation system. The generator module uses an RNN and the discriminator module uses a CNN-based architecture with one-dimensional convolutions over time. During the training, for a given music segment of a fixed size, the generator module generates movements frame-by-frame. Once a movement segment of the same size as the input generated, the discriminator takes both the input music segment and the whole generated movement to classify whether the movement is *real* or *fake*.

Weight Generation

A different approach in using neural networks to generate movement is to use a function to dynamically generate the weights of a regression neural network at each frame. The network is then used to generate the next mocap frame from the previous mocap frame as well as a set of control parameters. The weight generation function is responsible for modulating the temporality of movement and therefore it frees the regression network from model it. As a result, simple and compact fully-connected networks (as opposed to RNNs, CNNs, etc.) can be used as the regression network. This also avoids instabilities of RNNs while supporting real-time generation (unlike CNNs).

Holden et al. (2017) (39) use the periodic phase of locomotion as the underlying latent variable that defines the weight-generation function. The phase function is a cubic Catmull-Rom spline and our control points are used for the spline in the phase function. Each control point is a set of weights for the regression network. The phase function then allows for interpolating and producing the network weights for any phase value between the control points using the cubic Catmull-Rom spline function. The regression network is a three-layer

fully-connected network with 512 units each and ELU activations. The network is trained to take the previous mocap frame and the user control parameters as input and predict the current mocap frame, the phase change, and extra control parameters as its output.

Starke et al. (2019) (43) build on the framework by Holden et al. (2017) (39) by using a neural network to learn both cyclic and non-cyclic phase functions. This also allows the model to learn movements with different functional factors.

While both approaches by Holden et al. (2017) (39) and Starke et al. (2019) (43) use a global phase function for the whole body, Starke et al. (2020) (47) instead define the phase information locally for each body part. This allows for individual body parts to move ashyncronously, which is necessary for modelling more complex movements such as interacting with the ball and other players in a basketball scanrio as demonstrated by the authors.

2.5.5 Other Techniques

Other than the aforementioned mainstream machine learning models, a few studies use other techniques to learn and generate movement.

Li et al. (2002) (3) introduce a technique called *motion texture* for editing motion capture data using a two-level statistical model. This approach overcomes the limited ability of linear systems in capturing highly nonlinear complex movements by introducing a hierarchical approach for modelling non-linearities. A set of *motion textons*, representing movement primitives, is learned using a Linear Dynamic System (LDS) at the bottom level. The top-level model then learns the distribution of the textons using a transition matrix and thus captures the global dynamics of the movement, as an entire sequence.

Multilinear Independent Component Analysis (MICA) is a generalization of Independent Component Analysis (ICA) and N-model Singular Value Decomposition (SVD), which models higher order dependencies for each factor. Liu et al. (2011) (18) use MICA and decompose the training data into multiple factors. With the assumption that the factors are statistically independent, different states of the factors are arranged in a tensor. In this approach, time-warping is applied on the training data to achieve structurally similar movements. Furthermore, the dimensionality of the data is reduced using PCA.

2.6 Evaluation Methods

A review of the literature on movement generation systems reveals different methodologies and varying degrees of emphasis with respect to the evaluation of the system and the quality of its output movements. The evaluations range from verifying the performance of the system itself with respect to the ground truth data or other objectives (e.g., (Alemi et al., 2015; Holden et al., 2016; Lee et al., 2018)(29) (32) (40)) to comparing the approach with alternative approaches in the literature (e.g., (Levine et al., 2012; Fragkiadaki et al.,

2015; Jain et al., 2015; Martinez et al., 2017; Li et al., 2017; Pavllo et al., 2018; Starke et al., 2019, 2020) (25) (28) (30) (35) (38) (41) (43) (47)).

A large number of the publications provide no formal assessment and rely on the authors' informal inspection of a small subset of hand-picked movements, generated by the proposed system, and reported via 2D sketches or supplementary videos.

The systems that control the planning factors of the generated movements (e.g., pointing position, stride length, etc.) use the Root Mean Square Error (RMSE) between the generated data and the target as an indicator of the performance of their systems (Herzog and Krüger, 2009; Yamazaki et al., 2005; Starke et al., 2019, 2020) (13) (4) (43) (47). While this evaluation gives a good measure of how accurately the model is able to generate the movements that satisfy the given planning constraints, they do not measure the believability of the movements. For example, a generated movement for picking up an object might precisely put the hand in the location of the target object, but do so by introducing artifacts in the movement that render the movement unrealistic.

In another group of studies, the authors quantitatively evaluate the prediction precision of their system (Wang et al., 2007; Taylor and Hinton, 2009; Fragkiadaki et al., 2015; Martinez et al., 2017) (9) (14) (28) (35). A test segment as set aside, and a portion of it is fed as an input to the model, with the task of predicting (generating) the consecutive frames. The precision of the prediction is then quantitatively assessed by using the RMSE of the the generated movement and the ground truth data. As we discuss in more details in the discussion section, using the RMSE to evaluate a movement generation system fails to take into account the stochastic and creative nature of human movement.

A number of works conducted studies involving human subjects assessing the perceptual qualities of the generated and recorded movements (Chiu and Marsella, 2011b,a; Tilmanne et al., 2012; Alemi et al., 2015) (20) (21) (23) (29). The main motivation behind these studies is to evaluate the believability of the generated movement, as well as to validate the generation of the intended expressions. In such studies, human subjects are presented with one or more movement animations and are asked to either compare them, for example based on which one is more realistic, or to categorize, rate, or rank them based on a given metric such as the valence dimension of affect.

2.7 Summary and Discussion

In this section, we summarize our findings from reviewing the literature on machine-learning-based movement generation. First, we look at the types of movement that are modelled, as well as the movement factors that are used to control the movements being generated. We then discuss the limitations and challenges in acquiring training data, followed by summarizing the approaches to learn, generate, and control movement. Finally, we make the case for better evaluation methods for movement generation systems.

2.7.1 The Choice of Movements And Scenarios

Humans are able to perform a broad range of movements with intricate modulations that come from various factors such as the physical characteristics of the mover, her or his affective state, intentions, and plans. No computational model is yet capable of learning and generating all types of movements with every possible modulation. Research on movement generation therefore is done on relatively small and constrained subsets of all possible movements and scenarios.

Although in many of the reviewed literature it is not clearly stated why certain movements and scenarios were chosen for learning and generation, we point out a number of elements that might play a role in choosing what to model: meeting the demands of a certain application (e.g., the movement repertoire of a video game character), the simplicity or complexity of the movement pattern and thus different challenges in modelling them (e.g., modelling walking versus grand pas de chat), or the availability of the training data for certain movements (e.g., there are more training data for walking movements compared to the data available for writing with a pen on paper). In addition, the focus of a group of works is mainly on introducing a new machine learning model and generating movement is used to demonstrate the capabilities of the new model, as in the work of Taylor et al. (2006). In the following, we provide a brief discussion of different aspects of what has been chosen to be generated in the literature and the type of problems that needs to be addressed.

Scenarios - Table 2.1 and Table 2.2 provide some insights into the movement types and scenarios that are the subject of the reviewed works. Walking is arguably the most commonly modelled form of movement (Wang et al., 2006b; Taylor et al., 2006; Wang et al., 2007; Tilmanne and Dutoit, 2010; Kulić et al., 2011; Tilmanne et al., 2012, 2014; Alemi et al., 2015; Starke et al., 2019) (7) (8) (9) (16) (19) (23) (27) (29) (43). The prevalence of walking can be explained by the short and cyclic nature of walking patterns, the large availability of training data, and its application in video games. While most works generate arbitrary walks, a few have addressed the problem of character navigation, which requires the model to provide a way to control the direction of the movement such as minimizing the divergence of the generated path from a target path (Holden et al., 2016) (32), or continuously adjusting the heading direction of the character as movements are being generated (Alemi and Pasquier, 2017; Starke et al., 2019) (36) (43).

Dance movements are also explored in a number of works (Brand and Hertzmann, 2000; Li et al., 2002; Qu et al., 2008; Crnkovic-Friis and Crnkovic-Friis, 2016; Alemi et al., 2017; Li et al., 2017; Pettee et al., 2019; Sun et al., 2021) (1) (3) (12) (31) (37) (38) (45) (46). Dancing implies precise timing and positioning rules for different body parts, especially in the case of dancing with a partner. In most cases, the dance is accompanied by music and the choices of movements and their timings are influenced by the music, as well as by

the particular choreography of the dance. Addressing these challenges remain open to the research community.

A few studies consider modelling sports scenarios (Wang et al., 2006a; Qu et al., 2008; Wang et al., 2008; Matsubara et al., 2010; Levine et al., 2012; Li et al., 2017; Lee et al., 2018; Starke et al., 2020). (6) (12) (10) (15) (25) (38) (40) (47) Similar to the challenges described in modelling arm movements, certain sports movements are often expected to satisfy constraints and interactions with other movers or objects. For example, kicking the ball in its exact position in space or ducking at the right time and position to avoid being hit by an opponent. These criteria pose great challenges in using purely machine-learning based approaches. Recent works are starting to tackle these problems using deep neural networks (Lee et al., 2018) (40).

Diversity of Movements - The level of diversity in the training data also plays a role in the choices involved in building generative movement models. A machine learning model can be trained on a dataset that contains samples from the same form of movement with no variations across the function, planning, expression, or personal movement signature dimensions. This model learns a specific pattern and generates movements similar to that pattern. On the other hand, models that are trained on a dataset that contains movements that vary across one or more dimensions learn multiple patterns or different modulations of the same pattern. When it comes to generating new movements, only a few of these models provide ways to control the characteristics of the generated movements, which will be discussed in Section 2.7.7.

Creating models that can generate a diverse repertoire of movements has two requirements. First, a machine learning model that has the capacity of learning all such variations (e.g., neural networks versus HMMs), or using a hierarchical architecture that allow breaking down the training task into multiple subtasks. Second, a diverse training dataset. The recent availability of large datasets and computational power have allowed training models on a wider variety of movements than before (Martinez et al., 2017; Pavllo et al., 2018) (35) (41). Yet, large datasets such as Human 3.6M are tailored for human activity recognition use cases. They mostly contain variations across the functional dimension, corresponding to everyday movements, and may not contain variations of the same function in the planning or expressive dimensions needed for many generative applications.

Factors Used for Control - We now summarize and discuss the movement factors (Section 2.3.2) that are used for control.

Function: Controlling the functional factors of movement allows the user to choose the function (action) and ideally make transitions from one function to another. For example, one can ask the model to generate walking movements, followed by jumping over an obstacle, and and then grabbing an object. The majority of the systems only model a single function and only a few works address controlling the functional factors of movement (Wang et al., 2005; Kulić et al., 2011; Lee et al., 2018; Starke et al., 2019, 2020) (5) (19) (40) (43) (47).

The main challenge in modelling the functional factors comes from the broad variations in how different functions are executed (e.g., the differences between walking and shaking hands). Controlling the functional factors, compared to controlling the planning and expressive factors, requires employing machine learning models with higher learning capacities that can accommodate the larger repertoire of movement patterns, or designing hierarchical systems that consist of individual models for each function. Another challenge comes from the need to generate transitions from one function to another, which requires performing so in a plausible manner, even if samples of such transitions do not exist in the training data.

Planning: The execution of a given movement function can be planned by one or more planning factors. Considering walking as an example, one can plan the walk by specifying its trajectory or by setting the stride length. In modelling the planning factors, the reviewed works mainly investigate controlling the trajectories of the hands (Lin et al., 2008; Herzog et al., 2008; Herzog and Krüger, 2009) (11) (13), or the trajectory of the agent on the ground plane (Holden et al., 2016; Alemi and Pasquier, 2017; Wang et al., 2019; Starke et al., 2019; Ling et al., 2020) (32) (36) (44) (43) (48), as well as multiple limbs (Starke et al., 2020) (47).

The challenges in controlling planning factors often come from the need to satisfying the given constraints to a desired level of precision as defined by the plan. Depending on the application, the movement might have to follow an exact trajectory or stop at an exact location in space to follow the plan. In most cases, the plan can be described formally through a set of constraints, which allows calculating the error the agent is making with respect to the given constraints. The movement can then follow the plan by minimizing this error. The minimization can be done through an offline optimization process, as done by (Holden et al., 2016) (32), by learning to perform the movements that cause a reasonably small error, or by designing a sensorimotor loop for the agent, and feeding back the perceived error the agent has made from the target and adjusting the movements as they are being generated.

Expression: A variety of affective expressions can be conveyed by modulating human movement. While reviewing the literature, two issues come up in designing generative models that can control the expressive factors of the generated movements.

First, as opposed to the planning factors, there is no perfect execution pattern for an expression. One can express the same affective state through movements that differ in the way they are executed, which can be influenced by various factors such as the characteristics of the mover or her cultural background. Therefore, unlike the planning factors, one cannot directly measure expressive qualities of a movement and use such measurements to control the desired expressive modulation. Consequently, supervised machine learning techniques are the common approaches to control the expressive factors.

Second, to control the expressive factors, one has to, explicitly or implicitly, choose a method to describe the affective state or quality. Therefore, systems that support controlling

expressive factors differ on their choice of expressive factors and the way the factors are described. Although the majority of the studies use informally described walking gaits (e.g., chicken walk, drunk walk, etc.) as the expressive factor, some use categorical (Taubert et al., 2012; Samadani et al., 2013) (22) (26) and dimensional (Alemi et al., 2015) (29) representations of affect as the expressive factor.

Modelling Interactions - A relatively small, but growing subset of the studies look at modelling the interactions between the agent (mover) and its environment or the interaction of multiple moving agents.

With the higher capacity of the neural networks, more works are looking at training contact-aware models. In such models, the information about the points of contact between the agent and its environment (e.g., feet touching the ground or hands touching a ball) are used to train the model (Min and Chai, 2012; Levine et al., 2012; Herrmann et al., 2017; Holden et al., 2017; Wang et al., 2019; Starke et al., 2019, 2020) (24) (25) (34) (39) (44) (43) (47). Yet, modelling the interactions between two or more movers, such as hand shakes (Taubert et al., 2012) (22), are less explored.

There are two main challenges in modelling interactions. First, there is a lack of publicly available motion capture data of agent-agent and agent-object interactions. Second, interactions with an object or another agent introduce hard constraints that the generated movements have to satisfy, such as the exact timing and positioning of different body parts.

2.7.2 Training Data

Although the training data is one of the fundamental components of machine-learning-based solutions, the lack of available training data limits the scope of the problems that can be tackled. The field faces a number of challenges when it comes to finding a desired set of training data to address a particular research problem:

- There is a shortage of publicly available motion capture databases. The majority of motion capture data are owned by film and video game industries, or are captured by independent research groups that do not publish them to the public.
- As mentioned in Section 2.4.3, only a few publicly available databases are well curated towards particular research questions. New databases are needed to be curated to provide a wider variety of movements to support the problems described in Section 2.7.1. This includes training data that contain movements with variations across the five movement dimensions to allow for creating models that are able to generate such variations.
- To fully take advantage of the supervised and semi-supervised learning algorithms, the research community needs more annotated databases. Annotations allow for controlling movements based on meaningful factors, and supports creating generative models that integrate well with agent models.

- There is also a shortage of training data for scenarios in which two or more agents interact with each other, which is necessary to develop movement generation systems that address inter-agent communications as well as agent-object interactions.

- Another challenge in building a large training set is the different skeleton configuration that each database uses. Each database uses a different number of joints and bone proportions. As a result, one needs to re-target the skeletons from multiple datasets to a uniform target skeleton before being able to combine them, which is time and labor intensive. There has been some efforts in building automatic movement re-targetting models (Villegas et al., 2018; Aberman et al., 2020) and their effectiveness for creating a uniform movement database for movement generation application is yet to be evaluated.

2.7.3 Modular Learning

While in most reviewed works a single machine learning model is used to learn and generate the entire repertoire of the movements the system is expected to learn, a generative system might instead utilize a group of machine learning models working in connection with one another. Making the training process modular works by breaking down the structure of the movement data into smaller components and training different models on different, smaller segments of movement.

We describe two ways that one can break down the complexity of movements: 1) following the physical structure of human body, and 2) segmenting the time dimension.

We can group together the moving parts of the body in different ways in the context of motion capture data: all of the body joints together, separating the upper and lower body joints, grouping joints belonging to individual limbs (e.g., right arm, left leg), and finally considering a single body joint. A system can be designed to follow such groupings and assign different machine learning models to different parts of the data. For example, Sukhbaatar et al. (2011) use a two layer design: at the bottom layer, they train individual models for the right arm, left arm, right leg, left leg, and the trunk at the frame level. On the top layer, a CRBM is then used to coordinate the movements of individual limbs.

Another approach to break down the mocap data is to split the longer sequence of frames into smaller segments, and train each segment by a separate model. For example, an approach that is common among HMM-based works is to split a walking cycle into a few movement primitives (e.g., right leg lifted, ...), and use a separate HMM to lean each primitive. During the generation, the output of the HMMs are then concatenate in the right order.

2.7.4 Loss Function

In training artificial neural networks, the loss function directs what the model does and does not learn. In the reviewed works, it is common to use the Mean Square Error (MSE) between the generated joint rotations or positions and their ground truth counterparts. However, because of the highly variable nature of movement, using MSE imposes limitations on the generative and creative performance of the model. While the predicted movement might be *perceptually* plausible, it might diverge from the ground truth *numerically*. Further research is needed to devise more effective loss functions that consider the natural variabilities in human movement.

2.7.5 Modelling the Time Dimension

Movement unfolds through time and likewise, motion capture data are represented in the form of sequences of frames. The generative models are trained on these sequences and are expected to create new sequences, directly or indirectly. In the reviewed literature, modelling sequences is handled differently depending on what machine learning model is used: flattening the time dimension, as done in most of the dimensionality-reduction-based models; sequences as conditional inputs as in CRBM and FCRBM; recurrent connections in RNNs; and convolution over time in CNNs. A growing number of works use a separate module that determines phase of movements and generates the weights of a neural network. This approach decouples modelling the dynamics of movement from predicting the next frame, making it possible to use simple fully-connted networks that only learn to predict next frame from the previous frame (Holden et al., 2017; Starke et al., 2019, 2020) (39) (43) (47). Further investigation is needed to compare the pros and cons of each technique. The approach to modelling sequences also has an impact on the memory and time complexities of both the learning and generation algorithms. As a result, the length of the sequences that can be modelled is limited by available computational resources.

2.7.6 Generation Algorithms

The mechanics of the generation algorithms depend on the machine learning model. An algorithm that is suitable for one group of models may not work for others. As a result, it is not easy to compare the algorithms with on another. In the following, we summarize algorithms used in the literature for each family machine learning models.

Dimensionality reduction (DR) techniques generate movements by choosing a point in the DR space, and projecting the data from the DR space back into the mocap space. This process is often followed up by post-processing procedures to re-sample the data into a sequence of frames as the time dimension is often flattened in DR models.

HMM-based models use the Viterbi algorithm (Rabiner, 1989) to sample from the distribution learned by the HMM. In cases where more than one HMM is used for different

portions of the movement, first the desired order of the multiple HMMs is determined manually or by sampling from another model. Next, mocap frames are sampling from each HMM and concatenated to create the final sequence.

A group of neural networks including feed-forward nets, CRBM, FCRBM, and RNNs use iterative sampling. Such models are trained to predict the next frame from an input sequence of previous frames. By iteratively performing this sampling operation while shifting the input sequence to include the newly predicted frames, a sequence is generated.

Since the Convolutional Neural Networks (CNNs) use convolution over time, a whole sequence is generated at once, as opposed to the iterative sampling of other neural networks. To sample from CNNs in encoder-decoder architectures, only the decoder module is used. Given a value of the latent variable (code), a complete sequence is generated.

2.7.7 Control Techniques

One of the challenges in movement generation is controlling the qualities of the generated movements. As described in Section 2.3, the term *factor* refers to the sources of influence on movement and we call the domain of possible values for each factor the *factor space*. In Section 2.7.1, we discussed what factors are used to control the movement in the literature. In the following, we summarize the key techniques that are used to implement the control mechanisms.

Supervised versus Unsupervised Learning - Most works use a supervised learning approach, in which the model learns a mapping between labels, the previous frames, and the frame(s) to be generated. Supervised approaches have the benefit of allowing the researchers to explicitly convey to the model what factors they want to be controlled. However, the performance of the model depends on the quality of the labels. Another challenge in using supervised techniques is the lack of annotated mocap datasets as mentioned in Section 2.7.2.

In unsupervised techniques, the factors of variation in the training data are determined automatically, without any prior knowledge of what they might correspond to semantically. The result is often a low-dimensional representation of movement that could be interpreted by means of experimentation, e.g., (Brand and Hertzmann, 2000; Wang et al., 2007) ① ⑨.

Using unsupervised methods eliminates the need for labeled datasets. However, the discovered factors are not defined by the researchers and highly depend on the variations that exist in the training data, which may or may not directly correspond to semantically meaningful factors. While this can make it more challenging to design a system for a particular application that requires controlling certain factors, such unsupervised methods can be used as pre-trained models for creating supervised models.

Individual Models - The simplest way to create a control mechanism is to train a separate model for each point in the factor space and switch between models to control the generation. Each model is trained only on the data that correspond to that particular point, thus only imitating the same factor value.

Some studies combine the outputs of the individual models to create movements for other points in the factor space. For instance, Herzog et al. (2008) (13) and Herzog and Krüger (2009) (13) train individual Hidden Markov Models (HMMs) for learning pointing movements that vary in the position the hand points at. To generate a movement that points to a target position, a set of local HMMs with aiming positions closer to the target position are selected. Next, the parameters of a new HMM for the target position are determined by interpolating the chosen HMMs, and a new output is generated from the newly constructed HMM.

Similarly, Tilmanne et al. (2014) (27) train a Hidden Semi-Markov Model (HSMM) for each variation that exists in their training dataset. To generate movements for a given new point in the factor space, the model parameters of the individual HSMMs are interpolated or extrapolated.

In another study, Tilmanne et al. (2012) (23) train an HSMM on a large set of neutral walking sequences and use a linear regression transformation technique to adapt the parameters of the HSMM to a particular walking style. The adaptation algorithm uses the data from a small set of walks with that particular style.

Other models in this category learn one model per factor state, e.g., (Tilmanne and Dutoit, 2010) (16), or learn a range within the factor space using the same model, e.g., (Taylor et al., 2006) (8), but provide no method for controlling the generated movements.

Parametric Probability Distributions - One way to learn factor variations is to use a parametric probability distribution to model the data. In a parametric probability distribution, the mean of the distribution is a function of the factor(s). As a result, the value of the factor influences the mean of the distribution and thus controls the characteristics of the movements sampled from the distribution.

This method is commonly used among the HMM-based studies, e.g., (Herzog et al., 2008; Herzog and Krüger, 2009; Yamazaki et al., 2005) (13) (4). Wang et al. (2006a) (6) use Self-Organizing Mixture Network (SOMN) of parametric Gaussians to create a probabilistic mapping from the factor space to the pose space. In another approach, Wang et al. (2006b) (7) use parametric Gaussians to build Stylistic DTGs (Song et al., 2003).

Optimization - A number of optimization-based solutions are used to control the generation. Levine et al. (2012) create an interactive controller by defining the control problem as a Markov decision process (MDP). The near-optimal policy chooses the transitions in the latent space that satisfy the user input factors at each frame and is pre-computed using non-parametric approximate dynamic programming.

Maximum A Posteriori (MAP) is also used to control the generation process (Wang et al., 2008; Min and Chai, 2012; Herrmann et al., 2017; Wang et al., 2019) (10) (24) (34) (44). The MAP problem is solved through a combination of probablistic sampling and gradient-based optimization techniques in both online and offline manners.

Labels as Extra Model Input - Another technique to control what the model generates based on given labels is to feed the labels as extra inputs to the model alongside the training data. In this way, the model learns the correlations between the labels and the training data. During the generation, we can set the label inputs to our desired values and perform the sampling procedures as usual to control the generated data, e.g., (Taylor, 2009; Wang and Artières, 2017; Holden et al., 2017; Lee et al., 2018; Starke et al., 2019, 2020) (33) (39) (40) (43) (47).

Built-in Support for Control - A machine learning model can be designed in a way that provides a mechanism for a factor variable to control the characteristics of the generated movements through its internal connections. For instance, Factored Conditional Restricted Boltzmann Machine (FCRBM) uses a context variable that controls the behaviour of the network through gated connection between the observations and the latent variables (Taylor and Hinton, 2009) (14).

Holden et al. (2016) (32) use a feed-forward convolutional neural network dedicated to controlling the behaviour of another machine learning model that is trained on movement data. The control network learns a regression model from high-level parameters (factors) to the hidden layer of the main machine learning model. In this approach, a different control network is trained for different applications (e.g., controlling the walking direction versus controlling the affect), while the same main network is reused.

2.7.8 Machine Learning Family

As presented in Section 2.5, different families of machine learning models such as dimensionality reduction (DR), Gaussian Processes (GP), Hidden Markov Models (HMMs), and Artificial Neural Networks (ANNs), are used to learn and generate movement. While we point out the strengths and limitations of each family, we acknowledge that further investigations are needed to discuss which types of movements each machine learning model is capable of learning and generating. For example, while most reviewed works that are based on DR techniques model walking movements, one needs to apply the same approaches to other types of movement for comparison. This would be challenging since it is not always easy to replicate the approaches described in the literature.

DR techniques have two limitations in learning human movement. First, DR models map static poses to a DR space and do not explicitly consider the dynamics of movement. This is overcame by using a dynamical model such as LDS (Qu et al., 2008) (12) to model the temporal characteristics. Second, DR techniques rely on pre-processing steps such as sequence alignments and fixed-length representation of the data, which could require extensive manual labor or limit the variety of movements that can be modelled beyond short and cyclic movements such as locomotion (Tilmanne and Dutoit, 2010; Samadani et al., 2013) (16) (26).

The GP-based models (Section 2.5.2) have shown to generalize well from a relatively small training data (Wang et al., 2007) (9). This capability makes them suitable for applications in which one is interested in generating movements that there is no pre-existing or similar samples for them in the training set. However, GPs are not inherently dynamic models and do not capture the temporal structures of the data. This is overcome by integrating them with a dynamic model such as HMM (Taubert et al., 2012) (22), or by the introduction of dynamic GP models such as GPDM (Wang et al., 2008) (10) and MF-GPM (Wang et al., 2007) (9). Another limitation of the GP-based models is that they are computationally expensive to train and draw samples from, and they require maintaining the complete training dataset for generation, which make them less efficient for realtime and interactive applications.

HMMs (Section 2.5.3) were designed to learn and generate temporal data such as human speech. Unlike GP-based models, they do not require retaining the training data, and their generation algorithms can be used in real-time applications. However, the learning and expressive capacity of HMMs is limited. To keep the computational cost manageable, most HMMs are trained with the assumption that the data follows a first-order Markovian dependency, meaning that the next frame of data only depends on the current frame. While the first-order assumption might be sufficient in modelling short and cyclic movements such as walking, it has limitations in modelling more complex movements. One way to overcome this limitation, as applied in the literature, is to train a group of individual HMMs to capture different movement primitives or different factor variations as demonstrated in most HMM-based studies (Yamazaki et al., 2005; Herzog and Krüger, 2009; Tilmanne et al., 2012) (4) (13) (23). However, it is still possible to learn longer movements with a single HMM as demonstrated by Brand and Hertzmann (2000) (1).

There are a variety of artificial neural networks (Section 2.5.4) used for movement generation, each with different characteristics and applications. Shallow, perceptron-based neural networks can only learn basic movements with few degrees-of-freedom.

There are a number of benefits to use RBM-based models for movement generation. First, they are dynamic models and have more representation capacity than HMMs (Taylor, 2009). Second, compared to RNNs, it is easier to train a CRBM or an FCRBM on real-valued, sequential data such as mocap data. This is because RBMs do not suffer from vanishing or exploding gradients or dying-out outputs. Third, an architecture such as FCRBM is designed to allow a vector to change the behaviour of the network, making it suitable for controlling the generation. However, due to using a learning algorithm than networks such as RNNs and CNNs, it is harder to combine the RBM models with other types of neural network or train deeper networks.

Over the past few years Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have shown promising results in learning long-term dependencies well beyond a first-order Markovian assumption in sequential data. It is easier to train deeper

networks with RNNs or CNNs compared to RBMs. These properties make them suitable for learning movements that intrinsically have longer-term and highly non-linear dependencies, following complex hierarchical characteristics. For example, while a walking cycle can be modelled with a single-order Markov process, many dance pieces contain phrasings that need to be defined over longer windows of frames rather than a single past frame.

The main differences between using RNNs and CNNs for movement generation is the way they handle the time dimension. RNNs can be used to generate movements frame-by-frame and thus they are more suitable for real-time generation applications, while CNNs generate movements one segment at a time. Because of this, CNNs do not suffer from the problems that RNNs have in dealing with long sequences such as vanishing or exploding gradients and dying-out outputs as mentioned in Section 2.5.4.

2.7.9 Evaluations

As presented in Section 2.6, a major gap in the field of automatic movement generation is the lack of a widely-accepted evaluation procedure for the proposed systems.

Evaluation of movement generation systems is a challenging task. First, movement generation is highly function-dependent. Each generation system models only a subset of possible human movements, such as walking and running, or picking up an object. Such a model is therefore only capable of generating movements functionally similar to the ones it has seen in the training data and can be expected to be evaluated for those movements only. For example, a model that can successfully learn and generate walking movements may not perform well in learning and generating more complex movements such as dancing. This makes it difficult to compare alternative approaches to movement generation if they do not model the same set of movement functions.

Second, each system targets a specific application and has to be evaluated towards meeting the specifications of that application. For example, a model aiming for real-time generation has to be evaluated towards its space and time complexities, whereas an offline generation system might prefer better quality over faster generation. As another example, a system that supports controlled generation has to be evaluated based on its control abilities. As a result, not all systems can be evaluated in the same way.

Third, movement generation is a creative task, which requires a different evaluation approach than rational problem solving tasks. Although a movement generation system can be evaluated based on memorizing and regenerating the movement in the training data, thus using perfect recall as a measure of evaluation, one can evaluate the system based on its creativity and generalization capabilities. For example, evaluating the quality of the system's output in generating movements that do not exist in the training set.

In this section, we discuss the challenges we face in evaluating movement generation systems. We then highlight the lack of comparisons between alternative approaches and

make the case for building a stronger community for movement generation and its role in evaluations.

Evaluation of Generative Systems For any generative system, there are two dimensions that can be evaluated: 1) the performance of the software or the algorithm, and 2) the system's quality at its generative task (output). These evaluations can be exploratory, to identify any issues with the system or to determine the characteristics that can be measured in later evaluations, or they can be descriptive, to assess the quality of the system according to some standards, metrics, or requirements.

For evaluating the software and the algorithmic aspects of a generative system, we are mainly interested in the computational and memory (time and space) complexities. One can therefore use the common space and time complexity analysis to evaluate the performance of a system. This becomes more important if the system aims at performing real-time generation.

One can evaluate a movement generation systems' short-term and long-term prediction errors using MSE between the generated movements and the ground-truth data as a proxy to ensure that the model continues to generate plausible movements over time and the output does not drift to implausible and unrealistic movements.

Validating the creative quality of the system can be difficult. First, as opposed to rational problem solving, creative tasks are those for which there is no such a thing as a well-defined preference relation or utility measure. This is where most attempts at evaluating movement generation systems presented here fall short while using the mean squared error (MSE) between the generated movements and the ground truth data. Using MSE implies that there exist a single correct prediction and that is the one closest to the ground truth. However, in human movement, there exist many possible poses that can proceed from a given sequence of previous poses. This results in a *space* of correct and plausible poses rather than a single correct *instance*.

Another aspect of evaluating generative systems is to browse the variety of possible inputs and observe the system's output to make sure the system produces the correct output. For example, if we ask the model to point at a particular location in space, we are interested in making sure the system does generate movements that satisfy the given conditions for a variety of inputs. While this can easily be done if the conditions are quantified (such as locations in space), it is more challenging if we are evaluating in terms of more qualitative factors such as emotional states.

For the qualitative aspects of generative systems, evaluation studies involving human subjects are designed and conducted. In such studies, one has to consider that creative tasks can be subjective and the cultural biases, as well as any individual judgments in the evaluation of the outputs of a generative system as to be taken into account in analyzing the responses from human subjects.

Role of Rendering In order to visually evaluate the generated movements, first they have to be rendered through movements of a 3D character and placed in a 3D environment. The way the character and the environment are depicted can influence how human observers judge the quality and characteristics of the movement or identify artifacts such as foot sliding. For example, environmental elements such as lighting, drop shadows, ground-plane representation, and proper rendering of the shape of the feet can help better observe contacts between the feet and the floor. In addition, proper considerations have to be taken into account to not to bias the human observers through the shape, texture, colors, and the overall aesthetic of the character when the purpose of the evaluation is to identify certain qualities such as affect.

Replicability The majority of the reviewed systems are difficult to replicate. First, the training data for many studies are not provided to the public. Second, only a few studies have published the source codes for their experiments. As a result, only a few studies have provided a comparison between their approach and the alternative ones.

Objective comparisons between alternative approaches based on well-defined metrics, tasks, and applications can speed up the research and facilitate the innovations in any field. Some fields such as computer vision take advantage of the availability of widely accepted datasets that come with well-defined tasks and evaluation methods, such as the ImageNet (Deng et al., 2009) dataset and the ImageNet Large Scale Visual Recognition Challenge. While the field of movement generation lacks such datasets and challenges, the Movement and Computing (MOCO) community[4] which has been established in recent years can be a place for setting up such datasets and challenges by a group of interdisciplinary researchers and artists.

2.8 Conclusion

With the increasing demand for dynamic and interactive contents across various media, the need for automatic content generation is more apparent and movement animation is no exception. Meanwhile, the recent advancement in the field of machine learning and the promising results in the domains of audio, vision, and text, machine learning has shown to be a prominent choice for learning generative models of spatio-temporal data.

In this paper, we provided a review of the body of literature on using machine learning techniques and motion capture data for the purpose of movement animation generation. We argue that advances in this field lead to a variety of applications in the video game and film industries, as well as in art practices by providing a less expensive, faster, and more flexible way to create movement animation content both in offline and interactive scenarios.

[4] https://www.movementcomputing.org/

We point out a number of gaps in each aspect of the reviewed systems. Above all, we raise the need for high quality training datasets with diverse and well-curated contents that serve particular research questions. The availability of public-domain datasets, in conjunction with the rapid progress of the field of machine learning, will pave the way to create more powerful movement generation systems.

The works reviewed here have been published in a variety of different communities, depending on the field where the focus of the contributions were. Studies with focus on computer animation side of the research have been published in conferences and journals such as SIGGRAPH and ACM Transactions on Computer Graphics. Studies that focus on the affect-expressive movements are published in IEEE Transactions on Affective Computing and International Conference on Affective Computing and Intelligent Inter- action (ACII). Studies that contribute to the field of machine learning and use motion capture data have been published in machine learning venues such as international conference on machine learning and IEEE Transactions on Pattern Analysis and Machine Intelligence. While some fields such as computer music or computer graphics take advantage of strong specialized communities, ISMIR[5] and ACM SIGGRAPH[6] respectively, a specialized community for human movement and computation has only recently been emerged through the MOCO community.

Bibliography

K. Aberman, P. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen. 2020. Skeleton-Aware Networks for Deep Motion Retargeting. *ACM Transactions on Graphics* 39, 4 (July 2020).

ACCAD. 2019. Open Motion Data Project by The Ohio State University Advanced Computing Center for the Arts and Design. Retrieved 2019-03-19 from `https://accad.osu.edu/research/motion-lab/system-data`

Acclaim. 2013. ASF/AMC Formats Specifications. Retrieved 2019-03-19 from `http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html`

Adobe. 2020. The Mixamo Database. Retrieved 2020-09-01 from `https://mixamo.com/`

AffectMe. 2016. AffectME: Affective Multimodal Engagement. Retrieved 2019-03-19 from `http://web4.cs.ucl.ac.uk/uclic/people/n.berthouze/AffectME.html`

O. Alemi, J. Françoise, and P. Pasquier. 2017. GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks. Poster accepted to the Work-

[5] `http://www.ismir.net/`

[6] `http://www.siggraph.org`

shop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining - https://ml4creativity.mybluemix.net/.

O. Alemi, W. Li, and P. Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*. 442–448.

O. Alemi and P. Pasquier. 2017. WalkNet: A Neural-Network-Based Interactive Walking Controller. In *Proceedings of the 17th International Conference on Intelligent Virtual Agents (IVA) (Lecture Notes in Computer Science)*, Vol. 10498. Springer, 15–24.

I. Bartenieff and D. Lewis. 1980. *Body Movement: Coping With the Environment*. Routledge.

J. Bates. 1994. The role of emotion in believable agents. *Communications of the ACM* 37, 7 (1994), 122–125.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, 1171–1179.

C. M. Bishop. 1994. *Mixture density networks*. Technical Report. Aston University.

M. Brand and A. Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 183–192.

J.-P. Briot, G. Hadjeres, and F.-D. Pachet. 2019. Deep Learning Techniques for Music Generation – A Survey. *arXiv:1709.01620 [cs]* (Aug. 2019). http://arxiv.org/abs/1709.01620 arXiv: 1709.01620.

C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. Narayanan. 2008. IEMOCAP: interactive emotional dyadic motion capture database. *Language Resources and Evaluation* 42, 4 (2008), 335–359.

BVH. 2019. Biovision BVH Format Specification. Retrieved 2019-03-19 from http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html

J. Chai, J. K. Hodgins, J. Lee, P. S. A. Reitsma, and N. S. Pollard. 2002. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*. ACM, 491–500.

C. Chiu and S. Marsella. 2011a. A style controller for generating virtual human behaviors. In *The th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1023–1030.

C. Chiu and S. Marsella. 2011b. How to Train Your Avatar: A Data Driven Approach to Gesture Generation. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents*. Springer, 127–140.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1724–1734.

CMU. 2019. The Carnegie Melon University Motion Capture Database. Retrieved 2019-03-19 from http://mocap.cs.cmu.edu/

Cologne DB. 2019. The MoCap Database of TH Köln University of Applied Science. Retrieved 2019-03-19 from http://mocap.web.th-koeln.de/index.php

L. Crnkovic-Friis and L. Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. *CoRR* abs/1605.06921 (May 2016).

J. G. Cruz-Garza, Z. R. Hernandez, S. Nepaul, K. K. Bradley, and J. L. Contreras-Vidal. 2014. Neural decoding of expressive human movement from scalp electroencephalography (EEG). *Frontiers in Human Neuroscience* 8 (April 2014), 188.

A. Dainis. 2019. C3D Format Specification. Retrieved 2019-03-19 from https://www.c3d.org/pdf/C3D.pdf

DanceDB. 2019. Dance Motion Capture Database of the University of Cyprus. Retrieved 2019-03-19 from https://dancedb.cs.ucy.ac.cy

K. Davids, S. Bennett, and K. M. Newell. 2006. *Movement System Variability*. Human Kinetics, Inc.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

C. Ennis, R. McDonnell, C. Ennis, R. McDonnell, and C. O'sullivan. 2010. Seeing is Believing: Body Motion Dominates in Multisensory Conversations. *ACM Transactions on Graphics* 29, 4, Article 91 (July 2010), 9 pages.

N. Fourati and C. Pelachaud. 2014. Emilya: Emotional Body Expression in Daily Actions Database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. 26–31.

K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. 2015. Recurrent Network Models for Human Dynamics. (Aug. 2015). arXiv:1508.00271 https://arxiv.org/abs/1508.00271

M. Gales. 1998. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech & Language* 12, 2 (1998), 75 – 98.

T. Geijtenbeek and N. Pronost. 2012. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. *Computer Graphics Forum* 31, 8 (2012), 2492–2515.

I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc., 2672–2680.

F. S. Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48.

A. Graves. 2013. Generating Sequences With Recurrent Neural Networks. (Aug. 2013). arXiv:1308.0850

E. Herrmann, M. Manns, H. Du, S. Hosseini, and K. Fischer. 2017. Accelerating statistical human motion synthesis using space partitioning data structures: ccelerating statistical human motion synthesis. *Computer Animation and Virtual Worlds* 28, 6 (2017), e1780.

D. Herzog, V. Krueger, and D. Grest. 2008. Parametric Hidden Markov Models for Recognition and Synthesis of Movements. In *Proceedings of the British Machine Vision Conference*. 163–172.

D. Herzog and V. Krüger. 2009. Recognition and Synthesis of Human Movements by Parametric HMMs. *Statistical and Geometrical Approaches to Visual Motion Analysis* 5604 (Jan. 2009), 148–168.

G. E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation* 14, 8 (2002), 1771–1800.

G. E. Hinton, S. Osindero, and Y.-W. Teh. 2006. A fast learning algorithm for deep belief networks. *Neural Computation* 18, 7 (2006), 1527–1554.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 9 (Nov. 1997), 1735–1780.

D. Holden, T. Komura, and J. Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Transaction on Graphics* 36, 4, Article 42 (July 2017), 13 pages.

D. Holden, J. Saito, and T. Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics* 35, 4, Article 138 (July 2016), 11 pages.

D. Holden, J. Saito, T. Komura, and T. Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH ASIA 2015 Technical Briefs*. ACM, 18–4.

E. Hsu, J. Popović, and K. Pulli. 2005. Style translation for human motion. *ACM Transactions on Graphics* 24, 3 (2005), 1082–1089.

C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. 2014. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (2014), 1325–1339.

A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. 2015. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. *CoRR* abs/1511.05298 (2015). arXiv:1511.05298 http://arxiv.org/abs/1511.05298

I. T. Jolliffe. 2002. *Principal Component Analysis*. Springer.

M. Karg, A.-A. Samadani, R. Gorbet, K. Kuhnlenz, J. Hoey, and D. Kulic. 2013. Body Movements for Affective Expression: A Survey of Automatic Recognition and Generation. *IEEE Transactions on Affective Computing* 4, 4 (2013), 341–359.

D. P. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]* (May 2014). http://arxiv.org/abs/1312.6114 arXiv: 1312.6114.

J. Kivinen and C. Williams. 2012. Multiple Texture Boltzmann Machines. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, Vol. 22. 638–646.

L. Kovar, M. Gleicher, and F. Pighin. 2002. Motion Graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*. 473–482.

D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. 2011. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research* 31, 3 (2011), 16 pages.

T. Kwon and S. Y. Shin. 2005. Motion modeling for on-line locomotion synthesis. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)*. ACM, 29–38.

W. Lamb. 1965. *Posture and gesture: an introduction to the study of physical behaviour*. G. Duckworth.

J. Lasseter and J. Lasseter. 1987. *Principles of traditional animation applied to 3D computer animation.* Vol. 21. ACM.

K. Lee, S. Lee, and J. Lee. 2018. Interactive Character Animation by Learning Multi-objective Control. *ACM Trans. Graph.* 37, 6, Article 180 (Dec. 2018), 10 pages.

S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun. 2012. Continuous Character Control with Low-dimensional Embeddings. *ACM Transaction on Graphics* 31, 4, Article 28 (July 2012), 10 pages.

C. Li, Z. Zhang, W. Sun Lee, and G. Hee Lee. 2018. Convolutional Sequence to Sequence Model for Human Dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 5226–5234.

Y. Li, T. Wang, and H.-Y. Shum. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques.* ACM, 465–472.

Z. Li, Y. Zhou, S. Xiao, C. He, Z. Huang, and H. Li. 2017. Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis. *arXiv:1707.05363 [cs]* (July 2017). http://arxiv.org/abs/1707.05363 arXiv: 1707.05363.

C. Lin, W. Shiang, H. Rau, and J. Chen. 2008. Applying neural networks and optimization techniques to the simulation of human motion. In *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics.* 3194–3198.

H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne. 2020. Character controllers using motion VAEs. *ACM Transactions on Graphics* 39, 4, Article 40 (July 2020), 12 pages.

Z.-H. Ling, L. Deng, and D. Yu. 2013. Modeling Spectral Envelopes Using Restricted Boltzmann Machines and Deep Belief Networks for Statistical Parametric Speech Synthesis. *IEEE Transactions on Audio, Speech & Language Processing* 21, 10 (2013), 2129–2139.

G. Liu, M. Xu, Z. Pan, and A. Rhalibi. 2011. Human motion generation with multifactor models. *Computer Animation and Virtual Worlds* 22, 4 (2011), 351–359.

L. Liu and J. Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics* 37, 4 (July 2018), 142:1–142:14.

G. Lu, L.-K. Shark, G. Hall, and U. Zeshan. 2009. Dynamic Hand Gesture Tracking and Recognition for Real-Time Immersive Virtual Object Manipulation. In *Procedings of the 2009 International Conference on CyberWorlds.* 29–35.

Y. Ma, H. M. Paterson, and F. E. Pollick. 2006. A motion capture library for the study of identity, gender, and emotion perception from biological motion. *Behavior Research Methods* 38, 1 (2006), 134–141.

J. Martinez, M. J. Black, and J. Romero. 2017. On human motion prediction using recurrent neural networks. (May 2017). arXiv:1705.02445

T. Matsubara, S. Hyon, and J. Morimoto. 2010. Learning Stylistic Dynamic Movement Primitives from multiple demonstrations. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*. 1277–1283.

A. Menache. 2000. *Understanding Motion Capture for Computer Animation and Video Games* (1st ed.). Morgan Kaufmann Publishers Inc.

J. Min and J. Chai. 2012. Motion Graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics* 31, 6, Article 153 (Nov. 2012), 12 pages.

M. Mori, K. F. MacDorman, and N. Kageki. 2012. The Uncanny Valley [From the Field]. *IEEE Robotics & Automation Magazine* 19, 2 (2012), 98–100.

MovingStories. 2019. MoDa: The Open Source Movement Database. Retrieved 2019-03-19 from http://moda.movingstories.ca

H. Müller and D. Sternad. 2009. Motor Learning: Changes in the Structure of Variability in a Redundant Task. (2009), 439–456.

M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. 2007. *Documentation Mocap Database HDM05*. Technical Report CG-2007-2. Universität Bonn.

NUSDB. 2019. National University of Singapore Motion Capture Database. Retrieved 2019-03-19 from http://mocap.cs.sfu.ca/nusmocap.html

F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. 2013. Berkeley MHAD: A comprehensive Multimodal Human Action Database. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 53–60.

D. Pavllo, D. Grangier, and M. Auli. 2018. QuaterNet: A Quaternion-based Recurrent Model for Human Motion. *arXiv:1805.06485 [cs]* (May 2018). http://arxiv.org/abs/1805.06485 arXiv: 1805.06485.

K. Pearson. 1901. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572.

T. Pejsa and I. s. Pandzic. 2010. State of the Art in Example-Based Motion Synthesis for Virtual Characters in Interactive Applications. *Computer Graphics Forum* 29, 1 (2010), 202–226.

X. Pennec and J.-P. Thirion. 1997. A Framework for Uncertainty and Validation of 3-D Registration Methods Based on Points and Frames. *International Journal of Computer Vision* 25, 3 (1997), 203–229.

M. Pettee, C. Shimmin, D. Duhaime, and I. Vidrin. 2019. Beyond Imitation: Generative and Variational Choreography via Machine Learning. *CoRR* abs/1907.05297 (2019). arXiv:1907.05297 http://arxiv.org/abs/1907.05297

A. Pina, E. Cerezo, and F. J. Serón. 2000. Computer animation: from avatars to unrestricted autonomous actors (A survey on replication and modelling mechanisms). *Computers & Graphics* 24, 2 (2000), 297–311.

H. Qu, Z. Yu, X. Wang, and H.-S. Wong. 2008. Motion synthesis based on dimensionality reduction. In *Proceedings of the First IEEE International Conference on Ubi-Media Computing*. 237 –242.

L. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

C. E. Rasmussen and C. K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press.

C. Rose, M. F. Cohen, and B. Bodenheimer. 1998. Verbs and Adverbs: Multidimensional Motion Interpolation. *IEEE Computer Graphics and Applications* 18, 5 (1998), 32–40.

A.-A. Samadani, B. J. DeHart, K. Robinson, D. Kulić, E. Kubica, and R. Gorbet. 2011. A study of human performance in recognizing expressive hand movements. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.

A.-A. Samadani, E. Kubica, R. Gorbet, and D. Kulić. 2013. Perception and Generation of Affective Hand Movements. *International Journal of Social Robotics* 5, 1 (2013), 35–51.

S. Schaal, A. Ijspeert, and A. Billard. 2003. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences* 358, 1431 (2003), 537–547.

M. Schuster. 2000. Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks. In *Advances in Neural Information Processing Systems 12*. MIT Press, 589–595.

I. A. Serlin, M. R. Berger, and R. Bar-Sinai. 2007. Moving Through Conflict: Understanding Personal and Cultural Differences Through Movement Style. *Journal of Humanistic Psychology* 47, 3 (2007), 367–375.

A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. [n.d.]. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1010–1019.

K. Shoemake and K. Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 245–254.

L. Sigal and M. J. Black. 2006. *HumanEva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion*. Technical Report CS-06-08. Department of Computer Science Brown University.

P. Smolensky. 1986. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*. MIT Press, 194–281.

Y. Song, L. Goncalves, and P. Perona. 2003. Unsupervised Learning of Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 7 (2003), 814–827.

S. Starke, H. Zhang, T. Komura, and J. Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics* 38, 6, Article 209 (Nov. 2019), 14 pages.

S. Starke, Y. Zhao, T. Komura, and K. Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics* 39, 4, Article 54 (July 2020), 14 pages.

K. A. Studd and L. L. Cox. 2013. *Everybody Is a Body*. Dog Ear Publishing.

S. Sukhbaatar, T. Makino, K. Aihara, and T. Chikayama. 2011. Robust Generation of Dynamical Patterns in Human Motion by a Deep Belief Network. In *Proceedings of the 3rd Asian Conference on Machine Learning, ACML*. 231–246.

G. Sun, Y. Wong, Z. Cheng, M. S. Kankanhalli, W. Geng, and X. Li. 2021. DeepDance: Music-to-Dance Motion Choreography with Adversarial Learning. *IEEE Transactions on Multimedia* 23 (2021), 497–509.

L. M. Tanco and A. Hilton. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings of the Workshop on Human Motion*. 137–142.

T. Tang, J. Jia, and H. Mao. 2018. Dance with Melody: An LSTM-autoencoder Approach to Music-oriented Dance Synthesis. In *Proceedings of the 26th ACM international conference on Multimedia (MM '18)*. ACM, 1598–1606.

N. Taubert, A. Christensen, D. Endres, and M. A. Giese. 2012. Online simulation of emotional interactive behaviors with hierarchical Gaussian process dynamical models. In *Proceedings of the ACM Symposium on Applied Perception*. ACM, 25–32.

N. Taubert, D. Endres, A. Christensen, and M. A. Giese. 2011. Shaking Hands in Latent Space - Modeling Emotional Interactions with Gaussian Process Latent Variable Models. In *KI 2011: Advances in Artificial Intelligence*. Springer, 330–334.

G. W. Taylor. 2009. *Composable, distributed-state models for high-dimensional time series*. Ph.D. Dissertation. University of Toronto.

G. W. Taylor and G. E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, 1025–1032.

G. W. Taylor, G. E. Hinton, and S. Roweis. 2006. Modeling Human Motion Using Binary Latent Variables. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06)*. MIT Press, 1345–1352.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (Dec. 2000), 2319–2323.

J. B. Tenenbaum and W. T. Freeman. 2000. Separating Style and Content with Bilinear Models. *Neural Computation* 12, 6 (2000), pages 1247–1283.

J. Tilmanne, N. dAlessandro, M. Astrinaki, and T. Ravet. 2014. Exploration of a Stylistic Motion Space Through Realtime Synthesis. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications*. 803–809.

J. Tilmanne and T. Dutoit. 2010. Expressive gait synthesis using PCA and Gaussian modeling. In *Proceedings of the Third international conference on Motion in games*. Springer-Verlag, 363–374.

J. Tilmanne, A. Moinet, and T. Dutoit. 2012. Stylistic Gait Synthesis Based on Hidden Markov Models. *EURASIP Journal on Advances in Signal Processing* 2012, 1 (2012), 72.

J. Tilmanne, R. Sebbe, and T. Dutoit. 2008. A database for stylistic human gait modeling and synthesis. In *Proceedings of the eNTER-FACE'08 Workshop on Multimodal Interfaces*. 91–94.

K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. 2000. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 3. IEEE, 1315–1318.

B. Tomlinson. 2005. From linear to interactive animation: how autonomous characters change the process and product of animating. *Computers in Entertainment* 3, 1 (Jan. 2005), 20 pages.

N. F. Troje. 2002. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision* 2, 5 (09 2002), 2–2.

N. F. Troje. 2008. *Retrieving information from human movement patterns.* Oxford University Press, Chapter 12, 308–334.

U Penn DB. 2016. University of Pennsylvania - SIG Center for Computer Graphics Multi Modal Motion Capture Database. Retrieved 2019-03-19 from https://fling.seas.upenn.edu/~mocap/cgi-bin/Database.php

R. Villegas, J. Yang, D. Ceylan, and H. Lee. 2018. Neural Kinematic Networks for Unsupervised Motion Retargetting. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8639–8648. ISSN: 2575-7075.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 975–982.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2008. Gaussian Process Dynamical Models for Human Motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30, 2 (2008), 283–298.

Q. Wang and T. Artières. 2017. Motion Capture Synthesis with Adversarial Learning. In *Affective Computing and Intelligent Interaction*. Springer International Publishing, 467–470.

X. Wang, Q. Chen, and W. Wang. 2014. 3D Human Motion Editing and Synthesis: A Survey. *Computational and Mathematical Methods in Medicine* 2014, 3 (June 2014), 1–11.

Y. Wang, Z. Liu, and L. Zhou. 2006a. Key-styling: learning motion style for real-time synthesis of 3D animation. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 229–237.

Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. 2005. Learning hierarchical non-parametric hidden Markov model of human motion. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*. IEEE, 3315–3320.

Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. 2006b. Learning Style-directed Dynamics of Human Motion for Automatic Motion Synthesis. In *IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06*. 4428–4433.

Z. Wang, J. Chai, and S. Xia. 2019. Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2019), 14–28.

X. Wei, J. Min, and J. Chai. 2011. Physically valid statistical models for human motion generation. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 19:1–19:10.

S. Wold, K. Esbensen, and P. Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1 (Aug. 1987), 37–52.

S. Xia, C. Wang, J. Chai, and J. Hodgins. 2015. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics* 34, 4 (July 2015), 119–119:10.

J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai. 2009. Analysis of Speaker Adaptation Algorithms for HMM-Based Speech Synthesis and a Constrained SMAPLR Adaptation Algorithm. *IEEE Transactions on Audio, Speech, and Language Processing* 17, 1 (Jan 2009), 66–83.

T. Yamazaki, N. Niwase, J. Yamagishi, and T. Kobayashi. 2005. Human Walking Motion Synthesis Based on Multiple Regression Hidden Semi-Markov Model. In *Proceedings of the International Conference on Cyberworlds*. IEEE, 445–452.

H. Yin and N. M. Allinson. 2001. Self-organizing mixture networks for probability density estimation. *IEEE Transactions on Neural Networks* 12, 2 (March 2001), 405–411.

H. Zen, N. Braunschweiler, S. Buchholz, M. J. F. Gales, K. Knill, S. Krstulovic, and J. Latorre. 2012. Statistical Parametric Speech Synthesis Based on Speaker and Language Factorization. *IEEE Transactions on Audio, Speech & Language Processing* 20, 6 (2012), 1713–1724.

Chapter 3

Data and Tools for Human Movement Computing

3.1 Introduction

Human movement computation can involve multiple components such as data capture, data processing and analysis, feature extraction, and information pipelines. This chapter introduces the data, models, and libraries that we have created during this Ph.D. Although independently developed, the following projects share the common theme of movement computing and machine-learning research on movement data.

3.2 MoDa and Movement Data

High-quality data is an essential part of machine learning solutions and the broader area of movement computation. As mentioned in Section 2.4.3, there are several publicly available movement datasets to be used to train machine learning models. However, there was no dataset available that would fit the problems we addressed through our research questions 3 and 4 (**RQ 3** and **RQ 4**), presented in Chapters 5 to 8. Namely, we could not find a dataset that contained variations of affect based on a dimensional affect representation, and we also could not find a dataset that contained high quality synchronized music and dance data. As a result, we have captured a collection of high-quality motion capture data during this Ph.D. We first present the web-based frontend we have developed to publish the data and then describe the collected data.

3.2.1 MoDa Frontend

To make movement data accessible to the research community, we developed MoDa, a repository of movement data with a web-based frontend to find, visualize, and download the data. Beyond providing access to the data collected for the research presented in this thesis, MoDa hosts several other datasets that are collected by other researchers as a part of the MovingStories partnership. The MoDa implementation we present in this thesis is the second iteration of MoDa. The original implementation of MoDa (MovingStories, 2019) was online between 2015 and 2018. After encountering technical difficulties, we decided to develop a new frontend to provide more robust and easier access to the data. Throughout this thesis, by MoDa we refer to the new implementation unless stated otherwise. MoDa is available at `https://movehub.omid.al/`.

At the top level, MoDa is a repository of multiple datasets. Each dataset can contain zero or more collections. Each collection includes one or more data records. A record represents a movement instance that is captured. A record can contain multiple files. For example, a record of a dance movement can include a `BVH` file that contains the skeletal representation of movement, a `TRC` file containing the raw marker data, two `MP4` files representing the video recordings of the dance from two different angles, an `MP3` file containing the music, and a `Readme.md` file containing notes and extra information.

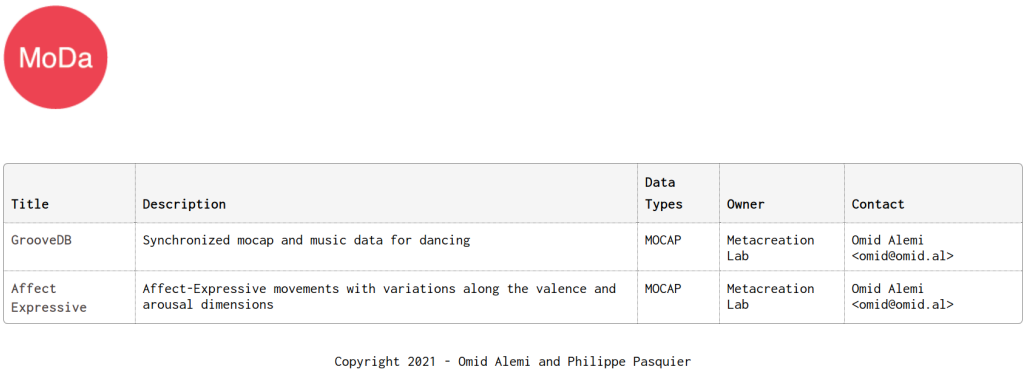


Figure 3.1: The homepage of MoDa, displaying the available datasets. From `https://movehub.omid.al/`.

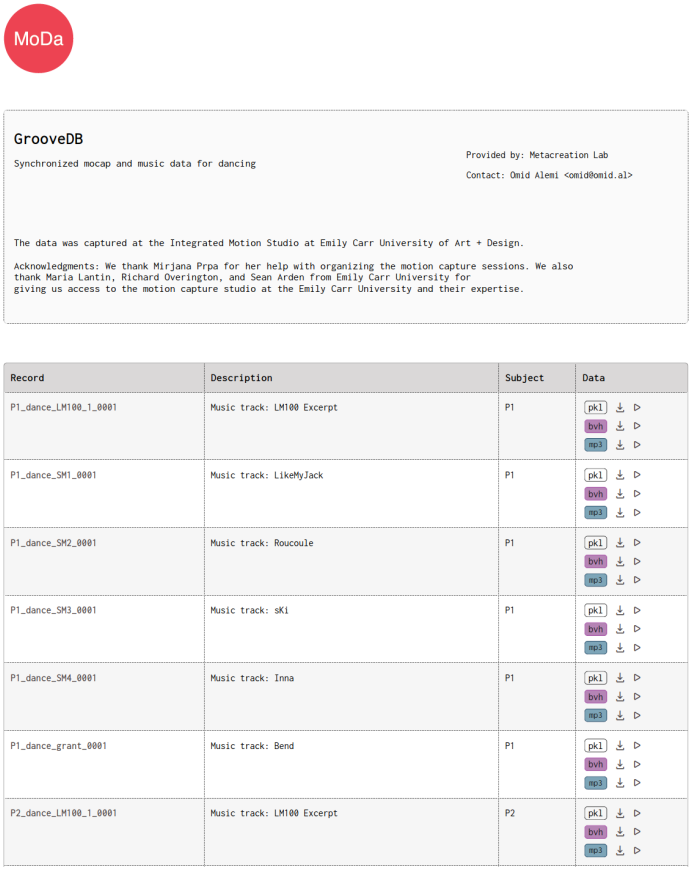


Figure 3.2: Multiple records within a data collection in MoDa.

A screenshot of MoDa's homepage is shown in Figure 3.1. Here, users can see the list of the available collections. As shown in Figure 3.2, The MoDa allows users to preview some file types such as motion capture data (only BVH), video, audio, images, and text files. Users can also download each file individually, all the files that belong to the same record, or all the files of a collection, grouped by their data type.

One of the motivations for the re-implementation of MoDa is to provide a faster and more robust way for users to access the data. MoDa should be online for years with no to minimal maintenance. To this end, we developed a static site generator for MoDa. Using static files, as opposed to relying on dynamic and server-side frameworks, allows us to host MoDa on a web-server that only serves static files, reducing the number of components and technologies that need to be secured and updated. In addition, serving static files supports caching the data and indexing by search engines.

Adding a new dataset to the MoDa repository is easy and flexible. The webpages are generated based on a given index file. MoDa does not make any assumptions on how the index file itself is created. For example, one can organize the data into folders and sub-folders, and run a script to scan the folders and create the index. As another example, one can create an index file based on a spreadsheet or data table. Examples of such scripts are provided in MoDa's source code as documented in Appendix B.

3.2.2 Affect-Expressive Movements Dataset

The first dataset we present is the *Affect-Expressive Movements Dataset*. The purpose of collecting this set of data is to provide motion capture data to train machine learning models that can recognize the affective state from movements (Li and Pasquier, 2016; Li et al., 2018) and train models that generate movement animation given the affective state (Alemi et al., 2015; Alemi and Pasquier, 2017). Accordingly, the data is designed to contain variations of affective state. To represent the affective state, we use a dimensional model of affect, defined along the two continuous axes of valence (V) and arousal (A) (Russell, 1980). This continuous representation is more suitable for learning a generalized model of affect than the categorical representations of affect. It allows us to predict or generate movements that correspond to affective states, not in the training data, and generate smooth transitions essential for interactive applications. To cover the full range of affective states while keeping the amount of the needed samples reasonable, we choose 9 points in the VA space to captures movements.

We captured the movements of 9 human subjects (five female, four male). As shown in Table 3.1, all nine subjects performed a walk along a figure-eight-shaped path. Two actors performed extra movements such as standing, walking in straight lines and taking sharp turns, sitting on a chair, and expressive arm gestures. Each movement is performed in 9 different modulations, corresponding to the nine regions in the circumplex model of affect, depicted in Figure 3.3. Each modulation differs in valence and arousal levels and is expressed

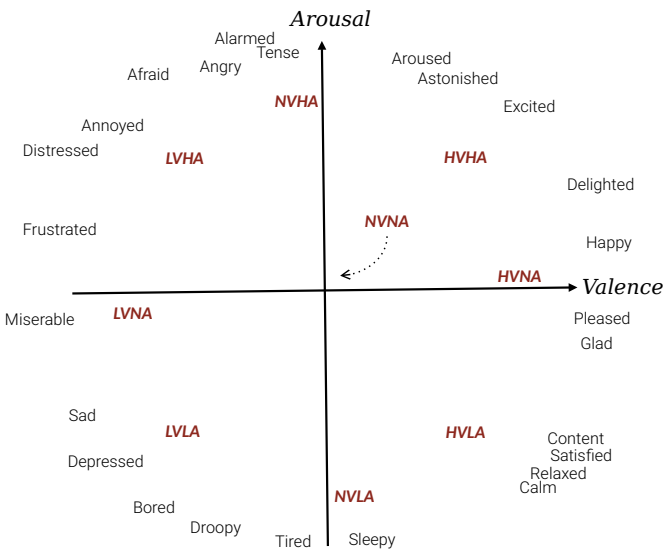


Figure 3.3: The circumplex model of affect. We record mocap samples from nine regions, highlighted by the acronyms in red, bold font: **H** stands for high, **L** stands for low, **N** stands for neutral, **V** stands for valence, and **A** stands for arousal. For example, ***HVNA*** means affective states with high valence and neutral arousal levels. The mapping to the categorical emotion labels is based on (Plutchik and Conte, 1997).

by full-body movements through mainly the body posture (its shape), the body parts' effort changes, and occasional arm gestures. The subjects perform each modulation multiple times to capture the natural variations in performing the same movement. We recorded the data in three iterations at the Integrated Motion Studio at Emily Carr University of Art + Design. We started the dataset by capturing two performers (P1 and P2) in early 2016, which we used for the development of AffectNet and WalkNet (Capters 5 and 6). We captured the third performer (P3) in late 2016, following by capturing the rest of the performers in 2020.

3.2.3 GrooveDB

The second dataset we present is the *GrooveDB*. The purpose of this dataset is to provide training data for cross-modal models of dancing to music. As a result, the dataset contains synchronized motion capture and music data. For each track, a performer dances to the music being played in a motion capture studio. The music tracks are in the genre of electronic dance music and are all computer-generated, and the dance moves are mostly grooving.

We recorded the data at the Integrated Motion Studio at Emily Carr University of Art + Design. The details of each track are shown in Table 3.2. To capture a variety of dance moves, we recruited ten performers in total (six female, four male). Seven of the performers had some form of dance background, while three did not have any dance training. All the performers were in their mid to late twenties. Photos from some moments of the performances are shown in Figure 3.4. We instructed the performers to groove with the

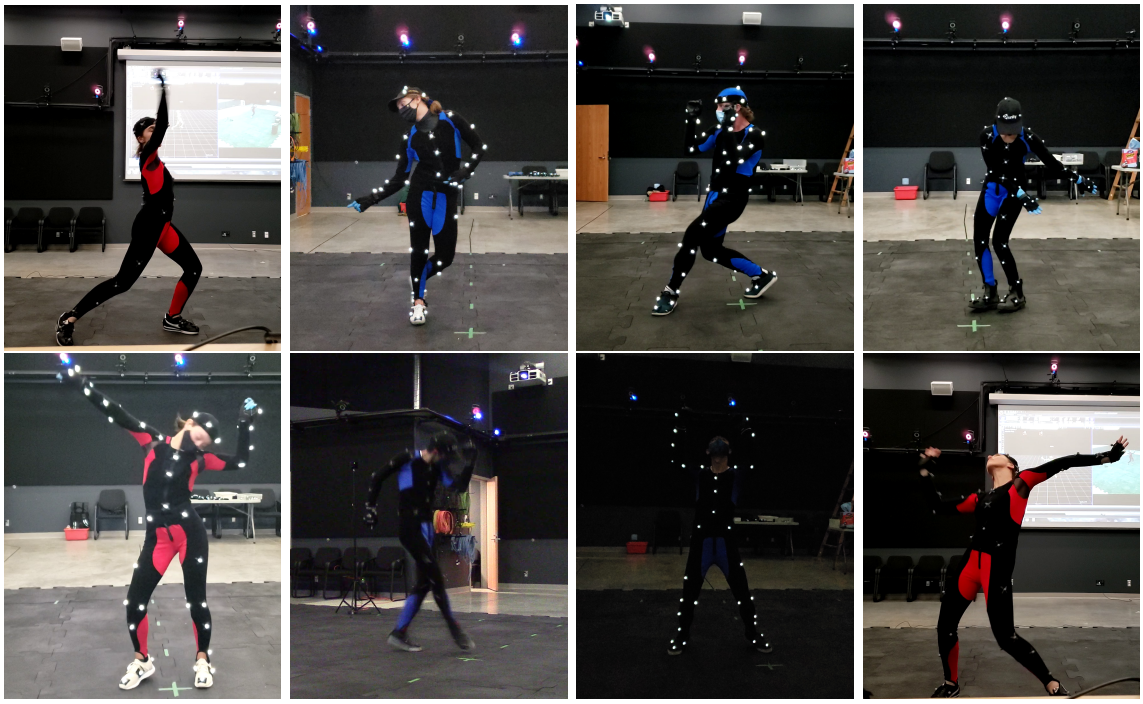


Figure 3.4: Photos from some of the performers in the GrooveDB

beats but did not give them any specific choreography to follow. Note that the duration of recordings for each song differs from performer to performer.

3.3 Mova: A Movement Analytics Platform

With the increasing interest in using data-driven approaches in human movement computation research, there is an increasing interest in analyzing, extracting, and representing human movements in terms of spatial, temporal, and qualitative characteristics. We can use information visualization techniques to help people better understand the contents of the movement data. To this end, we present the prototype of an interactive movement analytics framework, called Mova, for feature extraction, feature visualization, and analysis of human movement data. Integrated with a library of feature extraction methods, this platform can be used to analyze movement qualities and investigate the relationships between its characteristics. Also, Mova can be used to develop and validate new feature extraction methods with the help of parallel visualization of multiple features. We discuss test-cases in which Mova can be used and detail the road-map for its further development in Chapter 4.

Movement	Description	Movers	Total Length (hh:mm:ss)
Figure-Eight Shape	Walking along a figure-eight-shaped path	P1, P2, P3, P4, P5, P7, P8, P10	01:05:22
Straight Walks	Walking along a straight line and making sharp turns	P1, P2	47:47
Improvising	Subjects improvising their movements - with feet moving	P1, P2	20:59
Improvising - Static	Subjects improvising their movements without moving their feet	P1, P2	17:49
Expressive arm gestures	Two subjects expressing the intended affect through arm movements, facing each other	P1, P2	13:45
Sitting	Sitting on a chair	P1, P2	17:57
Pointing	Sitting on a chair and pointing	P1, P2	21:19
Hugging	Two subjects hugging each other	P1, P2	04:06
Lying Down	Lying down on the floor	P1, P2	19:43
			03:46:40

Table 3.1: List of Affect-Expressive Movement Data. The data is available at `https://movehub.omid.al/affectmove/`

3.4 m+m: Middleware for Movement-based Interactive Systems

m+m, a collaborative project lead by Thecla Schiphorst as the Principal Investigator and Ulysses Bernardet as the post-doctoral researcher, is an open-source software framework that facilitates building real-time, interactive systems that use movement data (Bernardet et al., 2016). m+m allows creating a graph of interconnected but independent components consisting of data acquisition modules, data processors, and effectors. m+m then handles the transfer of movement data between these components, whether they are located locally or distributed over a network. We contributed to m+m by integrating MoDa and Mova into the framework. For more details about the m+m project, please refer to Appendix E.

3.5 PyMO: A Library for Machine Learning Research on Motion Capture Data

As we started our research, the field of machine learning was undergoing a radical shift that resulted in new techniques such as deep learning models and new toolsets and libraries such as the shift from Matlab to Python for implementing machine learning algorithms and the introduction of machine learning libraries such as Theano, Tensorflow, and Torch. However, the tools and libraries for human movement computing were still behind these

Music Track	Movers	BPM	Music Length (mm:ss)	Captured Length (mm:ss)	Credits
LikeMyJack	P3 to P12	128	4:58	42:10	*
Roucoule	P3 to P12	128	7:02	32:27	*
sKi	P3 to P12	135	8:10	33:36	*
Inna	P4 to P12	126	6:07	31:36	*
Chasing the Learning Rate	7 to P12	125	3:07	18:44	§
Libra Mix 100 excerpt	P4 to P12	103	3:00	16:09	†
Libra Mix 97 excerpt	P8	132	1:30	1:30	††
Bend	P4 to P8 - P10 to P12	118	7:09	30:45	‡
				03:21:02	

Table 3.2: The contents of the GrooveDB. The data is available at https://movehub.omid.al/groovedb/. The *Music Length* column shows the length of the music track while the *Captured Length* column shows the total length of the captured data based on this music track from all of the performers. Note that the duration of recordings for each track differs from performer to performer. Music credits: *: Philippe Pasquier and Philippe Bertrand at the Robonom sound studio in France using the StyleMachine lite by Metacreative Technologies Inc. §: Philippe Pasquier and Renaud Bougueng Tchemeube at the Robonom sound studio in Vancouver using the MMM model (Ens and Pasquier, 2020) in Apollo (). †: By Doc Jay - From https://libramix.org/. ††: By Andry - From https://libramix.org/. ‡: By Grant.

changes. It was not easy to integrate movement processing libraries into more modern machine learning pipelines. In the beginning, we had to patch together multiple algorithms that were implemented in Matlab and Python. Therefore, we decided to develop PyMO to bridge this gap. PyMO is a Python library designed to support operations on motion capture data for machine learning applications. PyMO is an opensource project hosted on Github[1]. The long-term objective is to develop a community of users and contributors around PyMO. The main goals of PyMO are as follows:

- Be agnostic to a particular machine learning framework such as Tensorflow or PyTorch.
- Provide parsers for a variety of motion capture file formats.
- Provide data writers to save motion capture data into a variety formats.

[1] https://github.com/omimo/PyMO

- Support different representations of motion capture data (e.g., Euler angles and quaternions) and conversions between them.
- Provide a collection of pre and post-processing algorithms that are used in the literature.
- Provide means to analyze and visualize the motion capture data, including 2D and 3D animation of the skeletal data.
- Can be integrated into the common tools for machine learning researchers such as Jupyter Notebooks.[2] to better support exploratory analysis of movement data.

At the time of writing this thesis, PyMO supports reading and writing motion capture data in the `BVH` format and reading `C3D` files, while implementation of an `FBX` parser is underway. It supports Euler angles, exponential maps, and quaternions to represent joint rotations. The pre and post-processing algorithms currently includes the algorithms we use in our research.

Internally, PyMO stores its data in `Pandas`[3] dataframes. Using the Pandas library allows the data to be used by the tools and libraries familiar to many machine learning or data science researchers. PyMO also stores metadata such as the skeleton hierarchy (if one exists) or each joint or marker's name.

PyMO currently supports a number of visualization tools. The motivation behind these tools is to allow researchers to quickly see the data within the same tools and pipelines they use for their research without using external tools. Figure 3.5 shows the use of PyMO to visualize a single motion capture frame. To support rendering the motion capture data in 3D we developed a web-based library called MocapJS[4], which allows for embedding or sharing motion capture data on the web. Figure 3.6 shows an example of this 3D environment directly in a Jupyter Notebook. We also use MocapJS to stream the generate movements from a Python engine to a web-based render, as used in WalkNet and GrooveNet 2.0 demos.

3.6 Rank-Based Affect Estimation from Motion Capture Data

We present a model for estimating the perceived affect levels of full-body motion capture data. We use a pair-wise ranking approach instead of the more common rating approaches to distinguish between different affective states. In a rating approach, one chooses an absolute number with a predefined range to quantify an observation. In contrast, in a ranking approach, one chooses the relative differences between two given examples (Yannakakis and

[2] `https://jupyter.org/`

[3] `https://pandas.pydata.org/`

[4] `https://github.com/omimo/`

```
In [20]: draw_stickfigure3d(pos_data[6], 1510)
Out[20]: <matplotlib.axes._subplots.Axes3DSubplot at 0x7f05ec573490>
```

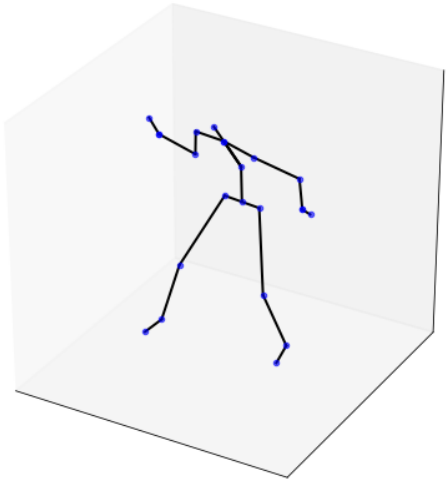


Figure 3.5: Visualizing a single motion capture frame in a Jupyter Notebook using PyMO

```
In [22]: nb_play_mocap('../groovenet-data/2020_raw/P4/P4_dance_SM3_0001.bvh')
Out[22]:
```

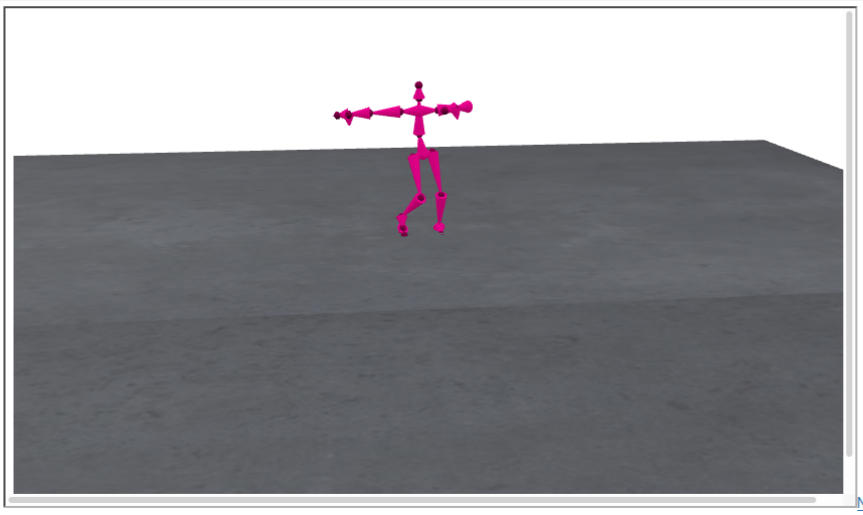


Figure 3.6: Rendering movement animation in 3D inside a Jupyter Notebook. This allows to quickly and directly animate a segment of the training data as well as generated movements represented in Python data structures without leaving the development environment.

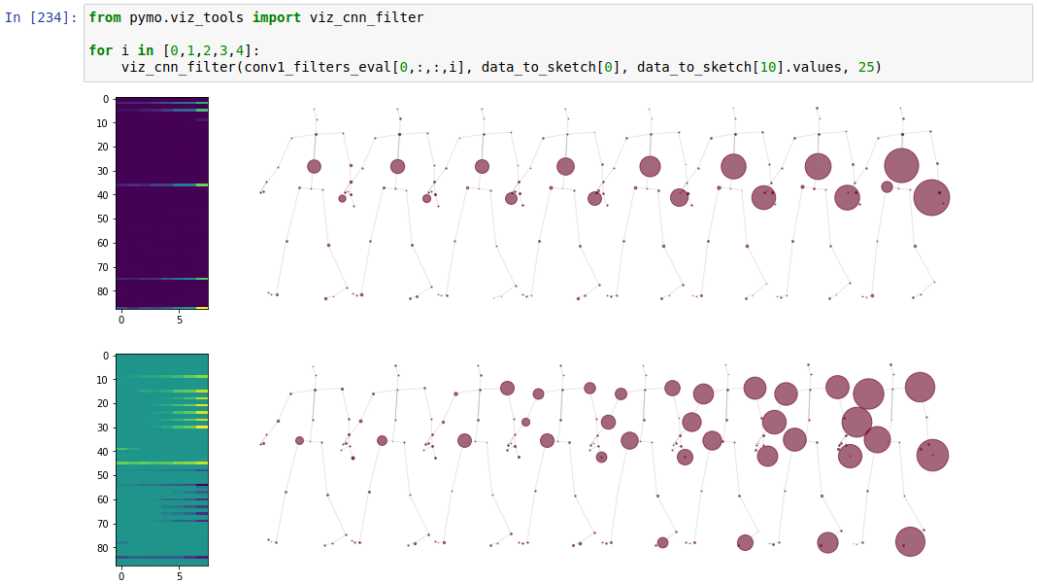


Figure 3.7: An example use-case of PyMO: Visualizing the movement features learned by a convolutional autoencoder. The autoencoder in this example is trained on a part of the affect expressive movements data (Section 3.2.2) and is tasked to learn a latent representation of the movements. This example shows how a movement visualization primitive (sketching in this case), can be used to analyze the features of a neural network, similar to feature visualizations of convolutional networks trained on 2D images.

Martínez, 2015). The benefit of the ranking approach is that, unlike the rating approaches, the participants do not need to understand the full range of possibilities. Also, Yannakakis and Martínez (2015) claim that ranking eliminates cultural and subjective biases and result in higher inter-participant agreements.

We use the affect-expressive motion capture data introduced in Section 3.2.2. We first use human participants to rank pairs of animated figures based on their perceived affect conveyed by the figures' movements. Using the resulting ranks as the ground-truth, we train a machine learning model based on the RankNet (Burges et al., 2005) model to predict the relative ranks for pairs of given motion capture clips. For more details about the ground-truthing study as well as the machine learning results, please refer to Appendix D.

3.7 Conclusion

In many applications, incorporating movement data includes several steps such as raw data acquisition from the sensors (motion capture, accelerometers, video, ...), processing the data through one or more modules, visualizing and analyzing the data, and possibly animating or sonifying the movements. Implementing each step requires technical expertise in different domains, such as programming, data acquisition, networking, machine learning, or visualization tools. Also, a pipeline has to manage the flow of data between each step.

As a result, it is difficult for individual researchers, artists, or developers to build interactive systems based on movement data. However, unlike the fields such as audio, music, image, or video analysis, which have multiple mature libraries and datasets, both the research community and the digital media industry lack well-established libraries for movement feature extraction and manipulation.

The works presented in this chapter are our attempt to bridge parts of this gap by developing tools that supported our research process and are provided to the research community as open-source projects. We conclude this chapter by highlighting the need for a library that provides robust, common implementations of the basic and advanced movement feature extraction and manipulation algorithms.

Bibliography

O. Alemi, W. Li, and P. Pasquier. 2015. Affect-Expressive Movement Deneration with Factored Conditional Restricted Boltzmann Machines. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 442–448.

O. Alemi and P. Pasquier. 2017. WalkNet: A Neural-Network-Based Interactive Walking Controller. In *Proceedings of the 17th International Conference on Intelligent Virtual Agents (IVA) (Lecture Notes in Computer Science)*, Vol. 10498. Springer, 15–24.

U. Bernardet, D. Adhia, N. Jaffe, J. Wang, M. Nixon, O. Alemi, J. Phillips, S. DiPaola, P. Pasquier, and T. Schiphorst. 2016. m+m: A Novel Middleware for Distributed, Movement Based Interactive Multimedia Systems. In *Proceedings of the 3rd International Symposium on Movement and Computing (MOCO '16)*. ACM, Article 21, 9 pages.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. ACM, 89–96.

J. Ens and P. Pasquier. 2020. MMM : Exploring Conditional Multi-Track Music Generation with the Transformer. arXiv:cs.SD/2008.06048

W. Li, O. Alemi, J. Fan, and P. Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In *Proceedings of the 5th International Conference on Movement and Computing (MOCO '18)*. ACM, Article 18.

W. Li and P. Pasquier. 2016. Automatic Affect Classification of Human Motion Capture Sequences in the Valence-Arousal Model. In *Proceedings of the 3rd International Symposium on Movement and Computing*. ACM, Article 15.

MovingStories. 2019. MoDa: The Open Source Movement Database. Retrieved 2019-03-19 from `http://moda.movingstories.ca`

R. Plutchik and H. R. Conte. 1997. *Circumplex Models of Personality and Emotions*. American Psychological Association.

J. A. Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology* 39, 6 (Dec. 1980), 1161–1178.

G. N. Yannakakis and H. P. Martínez. 2015. Ratings are Overrated! *Frontiers in ICT* 2 (2015), 13.

Chapter 4

Mova: Interactive Movement Analytics Platform

This chapter presents content from the following paper:

Omid Alemi, Philippe Pasquier, and Chris Shaw. 2014. Mova: Interactive Movement Analytics Platform. In Proceedings of the 2014 International Workshop on Movement and Computing (MOCO'14). ACM, 37–42. DOI: https://doi.org/10.1145/2617995.2618002

Abstract

There is an increasing interest in analyzing, extracting, and representing human movements in terms of a set of spatial, temporal, and qualitative characteristics for applications such as human-computer interactions and sports and health movement analysis. Information visualization techniques can be used to help people better understand the contents of movements. While all the characteristics of movement may not always be visible or detectable by humans, visualizations can illustrate detailed information about the characteristics of the movement. We present the prototype of an interactive movement analytics framework, called Mova, for feature extraction, feature visualization, and analysis of human movement data. Integrated with a library of feature extraction methods, this platform can be used to analyze movement qualities and investigate the relationships between its characteristics. In addition, Mova can be used to develop and validate new feature extraction methods with the help of parallel visualization of multiple features. We discuss test-cases in which Mova can be used and detail the road-map for its further development.

Link to the platform: `http://www.sfu.ca/~oalemi/mova/`

4.1 Introduction

Analyzing and understanding human movement has gained the attention of researchers especially in the fields such as human-computer interaction, computer animation, and sports and health research. Human movement is a form of non-verbal communication that conveys information about its performer (Troje, 2002, 2008). An important aspect of studying human movement is to extract and represent its content in terms of a set of spatial, temporal, and qualitative characteristics such as kinematics (speed, acceleration, and jerk), shape (contraction/expansion of body), the body structure (balance, center of mass, distance between body parts, etc), effort (space, time, weight, flow), emotions, and gestures. These characteristics are then used to interpret the movement and draw conclusions about it. For example, identify the functional, executional, and expressive aspects of human movement, recognize gestures and their qualities, index a movement database, or incorporate encoded movement features into a movement generation model in order to generate new movements based on a set of required features[1].

Movement information is represented in different forms from a simple video recording of the actor to the configuration of joints position (posture), joints acceleration, physiological properties of body parts, breathing, or the gaze of the actor. This information can be recorded from human actors through sensors (motion capture, accelerometer, etc.), or be generated using computational techniques. On the one hand, raw movement data provide little information about many underlying characteristics of the movement. Such characteristics need to be determined by human experts, analytical calculations, or machine learning approaches. On the other hand, once the characteristics are determined, a visual representation system is needed to convey this information to others.

Information visualization techniques can be used to illustrate movement and provide more information about its characteristics. Visualizations can be dynamic or static and can represent only the body skeleton or more high-level features of the movement. Movements can be visualized using different techniques ranging from the raw video recording of the actor/performer, the skeletal representation, animation, movement notation systems (e.g., Benesh or Labanotation) to more arbitrary visual representations of movement characteristics.

We present a prototype of *Mova*, an interactive movement analytics platform which is developed as part of Movement + Meaning Middleware project[2] that aims to provide a computation framework for movement information representation and manipulation. Mova is an open-source and web-based platform that integrates a set of extensible feature extraction methods with a visualization engine within an interactive environment. It seeks four goals:

[1] In this paper, we use the terms *characteristic* and *feature* interchangeably.

[2] `http://mnm.hplustech.com/`

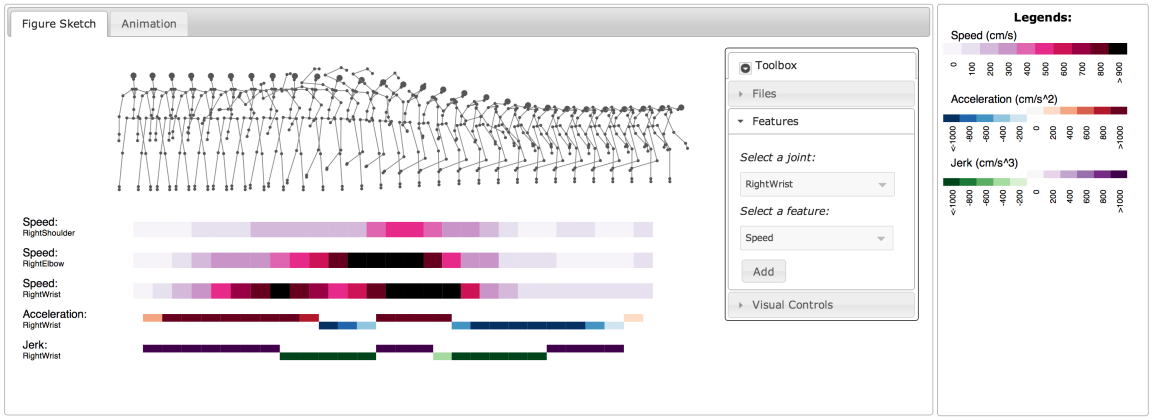


Figure 4.1: The main window of Mova visualizing kinematic features of a punch. The input data, as visualized by the stick figures, consists of a motion capture clip of a punch. A set of extracted features (speed, acceleration, and jerk) for three different joints are computed from the input data and visualized in the bottom.

(1) provide a platform for integrating and visualizing movement and its characteristics from a variety of data sources or representations (for example, video recordings, motion capture, accelerometer, and physiological be visualized together); (2) provide an analytical tool for researchers interested in analyzing movement characteristics of a human actor; (3) provide a research tool that can facilitate development and evaluation of movement feature extraction techniques; (4) be used for analyzing both the recorded movements of real humans and artificially generated movements of virtual characters.

The main contributions of this paper are as follows:

- It is a general purpose movement visualization and analysis platform which supports discrete and continuous features.
- It uses the parallel visual processing capabilities of human perception to visualize multiple features of the movement in parallel and in different forms which can be used to better understand the relationships between a particular class of movements and their corresponding measurable features.
- Mova is open-source and web-based which makes it suitable to be used in many applications including a front-end for movement databases where users can pre-view/examine the contents of each database entry.
- It is extensible: one can add new visualization techniques or feature extraction methods to the platform.

The prototype of Mova is accessible using the supplemented website mentioned in the abstract.

4.2 Background and Related Work

4.2.1 Visualizing Human Movement

Different techniques can be used to visualize movement and its characteristics. The raw movement data can be simply visualized using the video recording of the performer or through animating the skeletal information. There are notational systems such as the Benesh, Eshkol-Wachman, and Labanotation that are used in order to describe choreographies. For other movement characteristics, visualization techniques are rather arbitrary.

Works on human movement visualization can be divided into three groups in terms of their applications: (1) artistic visualizations, (2) movement summarizations, and (3) movement information analytics.

Artistic visualizations map the movement to aesthetic visual representations. They often target ordinary users rather than researchers and movement analysts and produce more abstract visualizations rather than analytical ones. For example, Bodycloud (Perret, 2009) creates sculptures from the resulting spaces of the movements of an actor. EMVIZ (flow) (Subyen et al., 2013) maps effort qualities to visual representation. through swarming boids.

Movement summarization visualizations are used to provide a synopsis or compare the contents of movement clips, often in a 2D space by projecting or eliminating the time dimension. For example, summarizing the contents of a large movement database allows its users to easily browse the contents. Furthermore, visualization techniques are also used in order to compare the similarities of a group of movement clips. 2D visualizations of movement clips are especially useful as they can be used in 2D media as well. Motion Belts (Yasuda et al., 2008) creates a timeline of 2D postures of key-frames of a clip. In this technique, the orientation of each body part is represented by a circular color scale. Motion Track (Hu et al., 2010) shows the variations within movement clips by first creating a 2D reference space of all key-frames within a database and then drawing a track for each given movement clip. As a result, similar movements would have tracks in the same areas of the space. Similarly, Motion Map (Sakamoto et al., 2004) depicts the trajectory of a movement in a reference space along with some key-poses across the space. Action Synopsis (Assa et al., 2005) technique illustrates short movements in still images by selected a set of representative key-poses that summarize the whole movement.

Movement information analytics provide further insights about the characteristics of the movement which are used in order to evaluate and understand a particular movement or a class of movements. Such visualizations are used by researchers, movement analysts, or choreographers in order to draw conclusions about the actor of the movement or the type of the movement he or she is performing. For example, ActionPlot (Carlson et al., 2011) is a visualization tool for contemporary dance that provides a structural analysis of dance performances through codifying the movements and providing more information about the performance. It visualizes the meanings, structural information, the performer's attention or

intention, the effort, the tempo of the effort, the balance of the movement, and the timing of actions being performed. Synchronous Objects (Palazzi and Shaw, 2013) uses visualization in order to provide choreographic structures of a performance. It is intended to provide a graphical language that can be used by researchers in an interdisciplinary environment to examine the relationships between space, structure, and movement.

4.2.2 Movement Feature Extraction

Movement features can be derived from any type of sensor systems that can record human movement such as video, motion capture, and accelerometers. Features are extracted in three ways: (1) using manual annotations determined by a human expert, (2) using algorithmic methods, and (3) using machine learning techniques.

Volpe (2003) suggest a set of motion descriptors and expressive cues such as Quantity of Motion (QoM), Contraction Index, Motion Strokes, Fluency, and Impulsiveness that are used in interactive multimedia systems for performing arts. Mentis and Johansson (2013) propose a set of rules to recognize Laban Movement Analysis (LMA) Effort parameters of the wrist joint. Kapadia et al. (2013) also propose a set of formulas to calculate Body, Effort, and Shape properties of a movement based on LMA. Such features are used to index a movement database which allows the user of the database to query movement clips using a set of desired feature values. EyesWeb (Camurri et al., 1999) extracts general space and gesture features from real-time video data. These features are used to determine the affect and emotion of the actors and are used to control sound, music, visual media, as well as to control actuators such as robots.

Machine learning is used for detecting features of movement instead of explicitly defining rules or algorithms. Gesture Follower (Bevilacqua et al., 2007) uses Hidden Markov Models to recognize gestures which, among other possible applications, is used for music pedagogy. Zhao and Badler (2005) use neural networks in order to extract Laban Effort qualities of live performances from both 3D motion capture and 2D video data. EffortDetect (Maranan et al., 2014) is a wearable system for real-time recognition of Laban Effort qualities using artificial neural networks.

Our work is a general-purpose platform that can be integrated with multiple feature extraction methods that were proposed in the literature in order to provide visualizations of such features for analytical purposes.

4.3 Movement Data

While designed to support multi-modal movement data in different representations, the current prototype of Mova only uses motion capture (mocap) data. In the future, other data representations will also be added to the system. Mocap systems use marker or marker-less techniques to capture the position and orientation of body joints in a 3D coordinate system.

These positions are mapped into a virtual skeleton and are often converted into a hierarchy of joint angle rotations to ensure the fixed limbs length.

Mova uses 3D position of the body joints for calculating the features and visualizing the skeleton. Mocap files in different formats (Biovision BVH, Acclaim, Collada, etc) are converted to an internal format in order to be used in Mova. The internal format which is used by Mova consists of the skeletal structure of the body and the position of each joint for every frame.

Mocap data are recorded with rates as high as 160 frames-per-second. However, in order to make the feature extractions computationally tractable, only a subset of frames is selected for manipulation. This selection can be done in two ways: (1) by sampling every k-th frame or (2) by extracting *key-frames*.

With the former approach, the user can specify the number of frames to skip which also defines the level of granularity of the features. The latter approach extracts a set of frames within the movement that, while sparse in time, are representative for the contents of the movement. The current prototype of the Mova uses the first approach. A future extension of Mova will include key-frame extraction capability.

4.4 System Design

Mova consists of three components: feature extraction components, visualization engine, and the graphical user interface (GUI).

The feature extraction component consists of a library of feature extraction methods. Each method analyzes the input movement data and returns the extracted values representing a feature for a specific body part or a group of body parts during the course of the movement as defined by the user. The features library also includes the information about the type of each feature (see Section 4.5) and the scale of the output values.

The visualization engine provides visual representations of the input movement clip and its extracted features. It particularly takes a parallel approach in visualizing the movement with multiple features: first, it provides multiple visualization techniques for the same feature and second it visualizes multiple features in the same screen. This allows the user to better observe the relationship of the features with one another and with the postures in the movement.

The GUI allows the user to load movement data, select the features and the joints to be visualized, choose between different types of visualizations, and control the properties of the system. The GUI also supports user interactions with the visualizations which enhances the efficiency and usability of the system.

4.5 Feature Extraction

Features are extracted from the changes of the body joints from one posture (frame) to another. These changes are determined from the position or rotation of body joints which are provided in the raw input mocap data. It is also possible to calculate features based on other features. For example, kinematics can be calculated directly from joint positions. Effort qualities are be derived from these kinematics features and the emotions of the actor can be further determined based on the effort qualities of movement. Mova calculates features from the movement data based the definitions described below:

Definition 1. *P_k^j represents the 3D position of the joint j at frame k.*

Definition 2. *R_k^j represents the rotation of the joint j at frame k.*

Definition 3. *Each frame (m_k) contains a set of positions (or rotations) for each joint defined in the virtual skeleton in the input data: $m_k = \{P_k^1, P_k^2, ..., P_k^G, \}$ where G is the total number of joints.*

Definition 4. *A movement clip is a set of consecutive frames with the length of N: $MOV = \{m_n | \forall n \in \{1, 2, .., N\}\}$.*

Definition 5. *In order to calculate movement features, a smaller subset of frames are selected from the original movement clip. The length of the selected frames set is much smaller than the length of the original movement clip: $SF = \{m_k | \forall k \in \{1, 2, .., K\}\}, K << N, SF \subset MOV$.*

Definition 6. *A movement feature describes a characteristic of the movement that is measured from the changes between two postures. The movement clip is then described by a set of feature values $\{f_1, f_2, ..., f_l\}$ which is calculated from the function g of the selected frames:*

$$g : SF \rightarrow \{f_1, f_2, ..., f_l\}$$

Definition 7. *Each feature value, f_i, is defined by:*

$$f_i = ([start_i, end_i), value_i)$$

The $start_i$ and end_i together represent the frame indices that define the window within the movement clip which the value is associated with.

Definition 8. *Based on the type the feature, the value can take three different forms:*

$$value = \begin{cases} r \in \mathbb{R} & (1) \\ s \in \{s_1, s_2, ..., s_n\}, & (2) \\ \emptyset & (3) \end{cases}$$

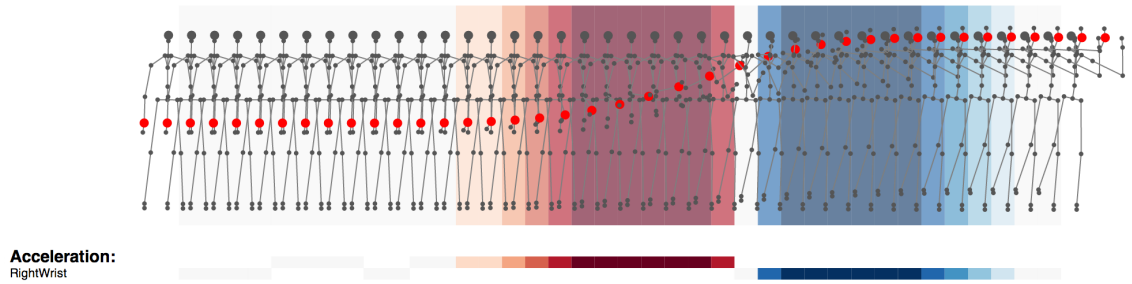


Figure 4.2: Figure Sketch for a sample movement with the values of acceleration of the right wrist projected on it.

(1) represents continuous real valued features such as speed or acceleration. (2) represents a finite set of discrete properties such as annotations. (3) represents features that only divide the movement into segments and thus only the start and end values of each segment are considered.

Example Features

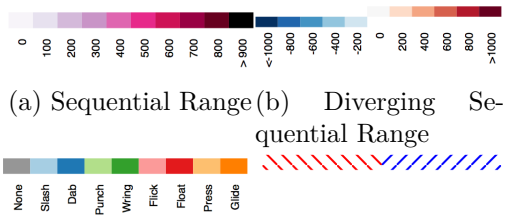
Below, is a list of movement characteristics that can be extracted and visualized in Mova using continuous, discrete, or segmented features. These features can be examined in terms of their functional, executional, or expressive qualities. Different approaches have been proposed in the literature in order to determine each feature and can be implemented in Mova (e.g., (Volpe, 2003; Kapadia et al., 2013; Mentis and Johansson, 2013; Maranan et al., 2014)).

- Kinematics: speed, acceleration, jerk.
- Laban effort parameters: space, time, weight, and flow
- Body shape measures (e.g., contraction/expansion of body)
- Gestures (e.g., swipe, drawing shapes, etc)
- Emotions (detecting emotions from posture and dynamic qualities of movement)
- Health conditions and gait patterns

  In order to test and demonstrate Mova, kinematics and Laban effort parameters are implemented in the prototype while the rest can be plugged-in to the system.

4.6 Movement Visualization

This section presents the design decisions and the techniques that are used in Mova for visualizing the movement and its features. The main window of the platform includes the



(a) Sequential Range (b) Diverging Sequential Range

(c) Annotations (d) Segments

Figure 4.3: Visualizing different types of features

tabs for accessing the Figure Sketch and the Animation visualizations, the features timeline, the legends, and the toolbox (Figure 4.1). We briefly describe each of these parts below.

Figure Sketch

Figure Sketch gives a general overview of the contents of the movement and its flow in a timeline in a 2D space (Figure 4.2). The visualization is done by drawing the postures from consecutive selected frames (from the SF set) within a short distance (controlled by a padding parameter) next to each other. The Figure Sketch can also be changed based on user interactions with other parts of the visualization.

Features Timeline

The features timeline represents the list of the features that are extracted from the movement. Each feature corresponds to a chosen joint by the user and is visualized as a timeline aligned with the postures on the Figure Sketech. This alignment allows the user to match each posture with the feature values that are calculated from the same time frame in the movement.

Depending on the type of the feature, a different visualization approach is used to illustrate the values:

- Real, continuous values that vary within a specified range are visualized using a multi-hued scale by varying the lightness of the colors proportionate to the value of the feature and the total range (Figure 4.3.a).
- Real, continuous values that vary within two diverging ranges centred on a pivot point (thus representing opposite polarities) are visualized using a distinct multi-hued scale for each polarity. The closer the value is to the pivot point, the lighter its color is. In addition, for better distinction of the ranges, the positive range is positioned higher than the baseline and the negative range is positioned below the baseline (Figure 4.3.b).
- Annotation are illustrated with a different color for each label (Figure 4.3.c). As annotations represent qualitative aspects of the movement and do not imply any

Figure 4.4: Drawing the trajectory of the right wrist joint colored based on the value of its speed while playing the animation.

order, the colors used for their visualization have different hues while their brightness are almost the same.

- Unlabelled segments that divide the movement clip into separate parts are visualized by the boundaries of the segments and an alternating fill pattern (Figure 4.3.d).

To get a better contrast and clarity in representing sequential data, we chose multi-hued color scales over single-hued scales. Multi-hued color scales use two or more hue levels in order to create a lightness gradient which makes it more difficult to design them. In order to automatically create good quality multi-hued scales Bezier interpolation and lightness correction (Aisch, 2013) are used (Figure 4.5).

In order to provide an easier perception of the relationship between the feature values and the state of the movement that they represent, upon hovering the mouse pointer over a feature timeline, an overlay of the feature values are projected behind the Figure Sketch (Figure 4.2). In addition, the joint which the feature is extracted from is highlighted.

Animation

The animation tab allows the user to play the movement as an animation. While animation is being played, the feature values that correspond to the current frame of the animation are highlighted.

The trajectory of a joint can be also drawn during the animation. The trajectory is visualized based on the value of a selected feature for that particular joint (Figure 4.4).

GUI Controls

Mova is an interactive platform which allows the user to manipulate the properties of the system though a toolbox. The toolbox includes controls for opening input files, selecting the joints and their features to be visualized, and changing the properties of the visualizations.

The file handling tool allows the user to upload new files or choose a previously uploaded file from the list to be visualized/analyzed. In addition, the user can save a snapshot of the current visualization in the form of a PDF or SVG graphic.

Figure 4.5: a) single hue vs b) multi-hued vs c) multi-hued with Bezier interpolation and lightness correction.

The feature selection tool allows the user to add new features to the features timeline by choosing a joint and a feature. Selecting the joint is made easy by displaying the skeleton to the user so that the user can click on the desired joint. Once added, the selected feature will be calculated for the specified joint and will be added to the timeline. The user can remove any of the features from the timeline once the feature is no longer needed.

The visual controls tool allows the user to change the visual properties of the system. The user can change the number of frames to sample for calculations (which is related to creating the *Selected Frames (SF)* set) which provides a form of *zooming* in terms of the granularity of the data being processed. It also allows the user to change the padding between each figure in the Figure Sketch which provides another form of *zooming* in terms of spacing the visualization at the pixel level.

4.7 Implementation

Mova is open-source and implemented from scratch as a web-based application in Javascript using open-source libraries. Its graphical user interface is built using *jQuery* library (The jQuery Foundation, 2013) and visualizations are implemented using *D3.js* library (Bostock, 2013). All the visualizations are implemented as Scalable Vector Graphics (SVG).

The features can be implemented in Javascript on the client-side or they can be hosted on a remote and more powerful machine and be called upon the need.

The complete source-code of the tool is available at `https://github.com/omimo/Mova`.

4.8 Test Cases

In this section we present three scenarios in which Mova can be used:

In the first scenario, a researcher implements a number of feature extraction methods for the same movement feature (for example, the Laban effort space quality) in order to compare and validate them. For doing so, the researcher can import annotated values of the feature that are determined by a certified movement analyst. Using the parallel visualization of each method, the researcher can easily notice which feature extraction methods comply with the expert opinion and thus validate or invalidate them.

In the second scenario Mova can be used in choreography/dance pedagogy or in heath movement analysis: a movement analyst extracts the features of a recorded movement clip from a dancer or a patient and analyzes the qualities of the movement in order to draw conclusions about the the performance of the dancer or the health conditions of the patient. In a similar way, one can evaluate the naturalness of the movements of a virtual character that are generated artificially by computer programs.

In the third scenario, Mova is used as a front-end of a movement database. Users can browse the contents of the database and open an entry of movement data (that is possibly recorded using multiple sensors) in order to examine its contents or evaluate its characteristics.

4.9 Conclusion and Future Work

We have introduced a prototype of an interactive platform for movement analytics which allows human movement experts and ordinary users to examine movement data in order to determine the underlying information and the characteristics they convey. The platform is interactive and allows its user to manipulate the visualizations. An important aspect of the platform is its use of the parallel visual processing capabilities of human perception in illustrating multiple features of the movement aligned with its postural representation.

Further developments of this prototype system include implementing:

- the support for combining multi-modal movement data such as video, mocap, accelerometers, breath sensors, and physiological sensors;
- a rich library of movement feature extraction methods proposed in the literature;
- automatic key-frame extraction;
- the support for features that involve a group of joints;
- the support for comparing the features of two or more movement clips;
- the support for inserting annotations to a movement clip;
- more variations of visualization techniques.

Acknowledgements

This work is supported by NSERC, SSHRC, and Canarie.

Bibliography

G. Aisch. 2013. Mastering Multi-hued Color Scales with Chroma.js. https://vis4.net/blog/posts/mastering-multi-hued-color-scales/ Accessed: 2013-12-15.

J. Assa, Y. Caspi, and D. Cohen-Or. 2005. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics* 24, 3 (2005), 667–676.

F. Bevilacqua, F. Guedy, N. Schnell, E. Flety, and N. Leroy. 2007. Wireless sensor interface and gesture-follower for music pedagogy. In *Proceedings of the 7th International Conference on New interfaces for Musical Expression, NIME*. 124–129.

M. Bostock. 2013. Data-Driven Documents (d3.js). http://d3js.org/ Accessed: 2013-12-15.

A. Camurri, M. Ricchetti, and R. Trocca. 1999. EyesWeb-toward gesture and affect recognition in dance/music interactive systems. In *IEEE International Conference on Multimedia Computing and Systems, 1999*, Vol. 1. 643–648.

K. Carlson, T. Schiphorst, and C. Shaw. 2011. ActionPlot: a visualization tool for contemporary dance analysis. In *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. 113–120.

Y. Hu, S. Wu, S. Xia, J. Fu, and W. Chen. 2010. Motion track: Visualizing variations of human motion data. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*. 153–160.

M. Kapadia, I. Chiang, T. Thomas, N. Badler, and J. Kider,Jr. 2013. Efficient motion retrieval in large motion databases. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '13)*. 19–28.

D. Maranan, S. Fdili Alaoui, P. Schiphorst, T. Pasquier, P. Subyen, and L. Bartram. 2014. Designing For Movement: Evaluating Computational Models using LMA Effort Qualities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. 991–1000.

H. Mentis and C. Johansson. 2013. Seeing Movement Qualities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, 3375–3384.

M. Palazzi and N. Shaw. 2013. Synchronous Objects for One Flat Thing. http://synchronousobjects.osu.edu Accessed: 2013-12-15.

R. Perret. 2009. *Bodycloud*. Master's Thesis. Zucher Hochschule der Kunste, Germany.

Y. Sakamoto, S. Kuriyama, and T. Kaneko. 2004. Motion Map: Image-based Retrieval and Segmentation of Motion Data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*. 259–266.

P. Subyen, T. Schiphorst, and P. Pasquier. 2013. EMVIZ(flow): An Artistic Tool for Visualizing Movement Quality. In *Proceedings of Electronic Visualisation and the Arts (EVA 2013)*. 32–39.

The jQuery Foundation. 2013. jQuery. `http://jquery.com/` Accessed: 2013-12-15.

N. F. Troje. 2002. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision* 2, 5 (09 2002), 2–2.

N. F. Troje. 2008. *Retrieving information from human movement patterns.* Oxford University Press, Chapter 12, 308–334.

G. Volpe. 2003. *Computational models of expressive gesture in multimedia systems.* PhD Thesis. InfoMus Lab, DIST–University of Genova.

H. Yasuda, R. Kaihara, S. Saito, and M. Nakajima. 2008. Motion Belts: Visualization of Human Motion Data on a Timeline. *IEICE - Transactions on Information Systems* E91-D, 4 (April 2008), 1159–1167.

L. Zhao and N. Badler. 2005. Acquiring and Validating Motion Qualities from Live Limb Gestures. *Graphical Models* 67, 1 (Jan. 2005), 1–16.

Chapter 5

AffectNet: Expressive Walking Movement Generation

This chapter presents content from the following paper:

Omid Alemi, William Li, and Philippe Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII). 442–448. DOI: https://doi.org/10.1109/ACII.2015.7344608

Abstract

The expressivity of virtual, animated agents plays an important role in their believability. While the planning and goal-oriented aspects of agent movements have been addressed in the literature extensively, expressing the emotional state of the agents in their movements is an open research problem. We present our interactive animated agent model with controllable affective movements. We have recorded a corpus of affect-expressive motion capture data of two actors, performing various movements, and annotated based on their arousal and valence levels. We train a Factored, Conditional Restricted Boltzmann Machine (FCRBM) with this corpus in order to capture and control the valence and arousal qualities of movement patterns. The agents are then able to control the emotional qualities of their movements through the FCRBM for any given combination of the valence and arousal. Our results show that the model is capable of controlling the arousal level of the synthesized movements, and to some extent their valence, through manually defining the level of valence and arousal of the agent, as well as making transitions from one state to the other. We validate the expressive abilities of the model through conducting an experiment where participants were asked to rate their perceived affective state for both the generated and recorded movements.

5.1 Introduction

Human movement is a form of non-verbal communication, which can be characterized along three dimensions: *function*, *execution*, and *expression*. The function dimension of a movement, at a cognitive level, defines the task that the movement is achieving, such as walking to a destination or picking up a cup from a table. This is taken into account by the means-end reasoning in virtual agent literature (Wooldridge, 2009). Note that for some movements such as dancing, the functional aspect may not be the most relevant characterization. The execution dimension of a movement reflects the pattern of the individual limb motions that constitute a movement. For example, walking is executed through locomotion or picking up a cup can be performed with either the *right* or the *left* arm's motion. The expressive dimension of movement represents the affective qualities that the movement is conveying, reflecting the emotional states felt or communicated by an agent or animated character. In computer animation, the expression of emotions is necessary for increasing the believability of virtual agents (Bates, 1994).

Agent animation can be created manually by animators, or computational models can be used to generate new animation automatically. The shift from linear (e.g., films and comics books) to non-linear (e.g., video games and interactive systems) media has increased the desire to build models for movement animation generation. Non-linear media requires a larger number of assets due to its dynamic and interactive nature. For movement animation, in particular, one needs to create variations of the same movement in order to respond to the need for a diverse set of movements performed in different forms and with different internal emotional states. However, creating a large number of assets manually is costly and time-consuming. Therefore, automatic generation can increase the efficiency in the production of such media. The motivation for automatic movement generation is two-fold: to serve as a computer-assisted creativity tool, and as the motor control for virtual agents.

Computer-Assisted Creativity: traditionally, animators use segments of recorded movements of real actors from a database of motion capture (mocap) data in order to create natural-looking movements. However, this method limits the movements to those that exist in the database, and recording all the possible variations of the same movement is not feasible. By using a generative model, animators can specify the characteristics of the movement segments they desire and the model would generate such movements, not limited to the existing set of movements.

Virtual Agent Movement: movements of an agent reflect its inner beliefs, goals, plans, as well as its affective state. While the literature has extensively addressed the relationship of the first three components with movement (Kovar et al., 2002; Shoulson et al., 2013), modelling the mapping between the affective state and the movement is still an open problem (Chi et al., 2000; Huang and Pelachaud, 2012).

We present an affect-expressive movement controller based on the generative capabilities of the Factored, Conditional Restricted Boltzmann Machines (FCRBMs) (Taylor and Hinton, 2009) trained on a corpus of motion capture data that is tailored specifically for this project. FCRBM has recently been applied to model the style of human movement such as the gait or the speed of the movement (Taylor and Hinton, 2009; Chiu and Marsella, 2011). In this paper, we extend the previous models by adding explicit control over the expressive qualities of the movement.

We have recorded the movements of two professional actors performing standing, walking, sitting on a chair, and expressive arm gestures. Following the model of valence and arousal (Russell, 1980) for representing affect, each movement type is performed with 9 different expressive combinations of the valence and arousal (Figure 6.1), which covers more emotional states than similar existing data sets (e.g., 4 emotions in the University of Glasgow's database (Ma et al., 2006)). Our choice of movements reflects our intention on building a generative movement model which synthesizes novel movement sequences, according to the goal-directed and the affective behaviour of an agent represented by a set of desired movement characterizations. Our corpus of movements adds emotional variations to those movements used by Motion-Graph-like structures (Kovar et al., 2002) that create streams of movements, making transitions from one type to the other based on a given set of requirements (e.g., following an arbitrary path). In this paper, we present the first stage of this model that addresses the expressive dimension of the agent movements, while adding the support for the function and execution dimensions is among the future direction of our work.

We annotate the mocap data based on their valence and arousal levels and train the FCRBM with its context variable set to the annotations during the learning procedure. Our intention is to use these 9 combinations of valence and arousal in order to build a generalized space of emotions, which allows us to induce any emotional state even if it is not within the original 9 combinations that were captured.

Experiments show that the model is capable of controlling the affective qualities of the generated movements through manually defining the level of valence and arousal of the agent. Furthermore, the model can interpolate and extrapolate between and beyond any two points in the affect space and generalizes well to unseen combinations. This feature can be used to create smooth transitions between two affective states or exaggerate certain states. We validate the ability of the model to convey the intended affective states through an experiment where human observers rated their perceived valence and arousal levels from both the recorded and generated movements. Note that the levels of the valence and arousal on the labels of the training data reflect the emotional states that were instructed to the actors. As the instructed or felt emotions might differ from the emotions felt and perceived by independent observers, we also collect and study the perceived emotions, which can be used as the ground truth for future experiments.

The rest of the paper is organized as follows: Section 5.2 reviews the related work in statistical movement generation. Section 5.3 explains the background of the machine learning model we use. Section 5.4 outlines our approach to model affect and our design decisions in the choice of movements for the training data set. Section 5.5 presents the results of our model. Finally, Section 5.6 summarizes the paper and outlines the future directions.

5.2 Related Work

Approaches to generate movement animation can be divided into physics-based and data-driven categories. While physics-based methods can successfully generate physically-valid and robust movements, it is challenging to capture the expressive qualities of movement using physical simulation. Data-driven methods, on the other hand, use pre-recorded movement data of real human actors and thus can better capture the expressive qualities of the movement that are visible in the data.

Data-driven movement generation has been approached by interpolating two movement sequences (Kwon and Shin, 2005), by concatenating short movement clips to make longer, functional movements (Tanco and Hilton, 2000; Kovar et al., 2002), and by using statistical and machine learning models. While the first two techniques are mostly limited to the movements that are available in the recorded data or their combinations, machine learning models are capable of generalizing movement qualities and generating novel movements.

Hidden Markov Models (HMMs) are used to generate human movement: the style machine extracts the stylistic variations of movements in an unsupervised manner and controls the generation using a set of stylistic degrees of freedom variables (Brand and Hertzmann, 2000). Wang et al. (Wang et al., 2006) use HMM with mixtures of Stylized Decomposable Triangulated Graph (SDTG) as the probability distribution of its visible states in order to model movement using a supervised style variable. Hidden Semi-Markov Models (HSMM) are used to parameterize the movement pace as well as its style (Niwase et al., 2005). Another study models expressive gaits during walking by training an HMM on a reduced-dimension space derived using Principal Component Analysis (Tilmanne and Dutoit, 2010).

Gaussian process models are also used in order to separate the stylistic characteristics of movements from their content. Multifactor Gaussian Process Models (Wang et al., 2007) are able to generate movements with stylistic variations (walking and running) that do not exist in the training data by learning those variations from other types of movement.

Extensions to Restricted Boltzmann Machines (RBMs) have been recently applied to model human movement. Conditional RBM (Taylor et al., 2006) is used to generate human movement. The Factored Conditional RBM (FCRBM)(Taylor and Hinton, 2009) extends the CRBM and includes a context unit which modulates the interactions between the hidden and visible units as well as the units from past time steps, which allows for controlling

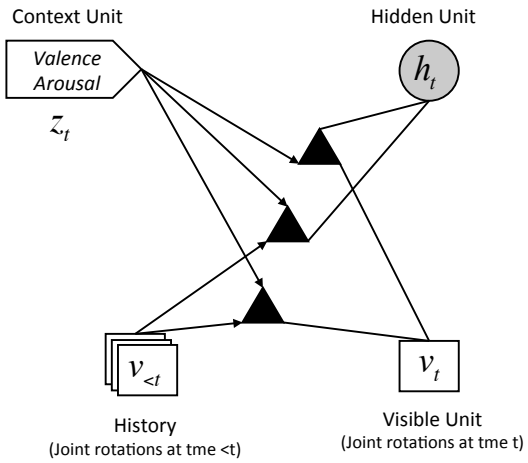


Figure 5.1: FCRBM's architecture with valence and arousal labels modulating the interactions between the past visible, current visible, and current hidden units.

the variations of movements while sampling new sequences. In another work (Chiu and Marsella, 2011), the authors propose a two-layer model called Hierarchical Factored Conditional Restricted Boltzmann Machine (HFCRBM) for learning and interpolating movement style using the middle hidden layer.

The works mentioned above only model movements based on their functional aspect or based on arbitrary expressive variations (e.g., chicken walk vs dinosaur walk). Furthermore, simply choosing between discrete *categories* of gait style does not provide a flexible method of controlling the expressivity of the characters or to producing the transitions from one emotional state to the other. To the best of our knowledge, no previous study has addressed building a generative machine learning model that allows controlling the expressive qualities of movement using a set of semantically valid variables such as valence and arousal of the affective state of a character.

5.3 Machine Learning Background

Factored, Conditional Restricted Boltzmann Machine (Taylor and Hinton, 2009) is an energy-based machine learning model for capturing the contextual information of time-series data. FCRBM, as shown in Figure 5.1, consists of a set of visible units, which represent the output at the current time-step, a set of past visible units which represent the history of the output, and a set of hidden units that represent a non-linear interpretation of the output and its past in order to learn the temporal patterns of the training data. It also uses a context unit which controls the interactions between each pair of units. By setting the context unit to the annotation values, the energy landscape of the model changes which

allows the model to learn the relationship between the contextual information provided by the annotations and the weights of the connections between the units in an efficient manner.

We use the FCRBM in modelling the expressive qualities of movement as it has a number of advantages over other approaches (Taylor, 2009). First, the hidden states of an FCRBM provide more representational power over HMMs. Second, using a feature variable, FCRBM provides the ability to control the characteristics of generated samples. Third, unlike the Gaussian process models, FCRBM does not need the training data set while sampling new sequences (except a few frames for initializing the model).

The multiplicative, three-way interactions make the parameters of the model cubic. However, by factoring the weight tensors into a product of pair-wise interactions (factors), we can approximate the weight tensor and reduce the order of the parameters to $O(N^2)$. This results in an energy function of the following form:

$$\begin{aligned} E(v_t, h_t|v_{<t}, z_t) = \frac{1}{2}\sum_i (v_{i,t} - \hat{a}_{i,t})^2 - \\ \sum_f \sum_{ijl} W^v_{if} W^h_{jf} W^z_{lf} v_{i,t} h_{j,t} z_{l,t} - \sum \hat{b}_{j,t} h_{j,t} \end{aligned} \tag{5.1}$$

where v_t and h_t are the visible and hidden units at the current time-step, $v_{<t}$ represents the past visible units, z_t represents the context information (labels) at the current time-step, f represents the index of the factors, $\hat{a}_{i,t}$ and $\hat{b}_{j,t}$ represent the dynamic biases of the visible and hidden units respectively, and W denotes the weight matrix between each unit and a factor. The model can be trained using the Contrastive Divergence algorithm and new samples can be generated by performing alternating Gibbs sampling. For more detailed explanation of the algorithms, refer to the work of Taylor and Hinton (Taylor and Hinton, 2009).

5.4 Affect-Expressive Movement

5.4.1 Affect Representation

Affective states are typically represented using categorical or dimensional models (Karg et al., 2013). Categorical representations define emotions using a set of labels that come from the everyday language uses (Tracy and Randles, 2011). Examples are anger, happiness, sadness, surprise, disgust, and fear. Dimensional models break down the affective states into two or more factors, which are represented as a point within a space defined by those factors (Plutchik and Conte, 1997). The most common example is the PAD model of affect, which defines emotional states based on arousal, valence, and in case of social situations, dominance (Mehrabian, 1996).

We use the arousal and valence dimensions (shown in Figure 6.1) in order to describe the affective state of movements, as they define the affective state with two degrees of

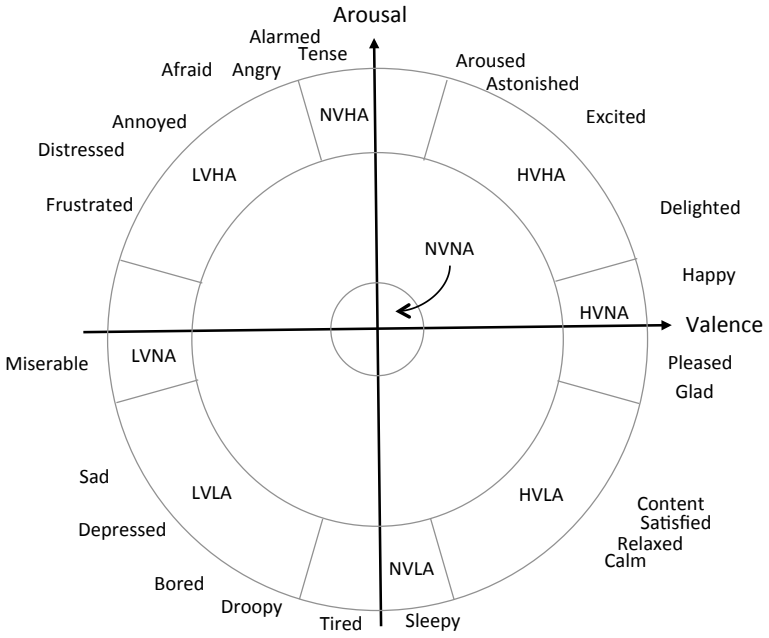


Figure 5.2: The affect model described by valence and arousal dimensions with the 9 zones recorded in the training data. The mapping to the categorical emotion labels are based on (Plutchik and Conte, 1997).

freedom, each across a continuum. This makes it more suitable for learning a generalized model of affect than the categorical models and allows us to create smooth transitions that are essential for interactive applications. In future, we plan to explore other affect dimensions (Fontaine et al., 2007). For example, using dominance to model the interactive and multi-agent scenarios.

5.4.2 Data Gathering

We have captured the movements of two professional actors (one female, one male). The actors were asked to perform standing, walking in different directions, and sitting on a chair as well as expressive arm gestures. These specific movements are chosen in order to be used to build a generative movement model that is capable of synthesizing movements, both on-line and off-line, given any arbitrary set of movement characterizations along the three dimensions of function, execution, and expression. In this paper, we only address the expression dimension. All of the recorded motion capture data and the reference videos are available at *http://moda.movingstories.ca/projects/22-affective-motion-graph*.

Each movement sequence is performed in 9 different expressive modulations, as indicated in Figure 6.1. Low, neutral, and high levels of valence and arousal are considered. Each modulation of the emotions is expressed by full body movements through mainly the body posture (its shape), the body parts' effort changes, and occasional arm gestures. Each

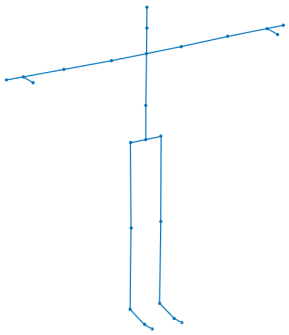


Figure 5.3: The skeleton used for the training data

modulation was also repeated 4 times in order to increase the variability of the training data.

Before training, we annotate each sequence using a two-dimensional variable representing their valence and arousal levels. The variable uses a continuous interval to represent the low, neutral, and high levels with the values of 1, 2, and 3 respectively. We decided to use this specific range of numbers after performing experiments with multiple ranges. Note that the values cannot be zero as such condition can cancel the weights in the model. Although the chosen modulations in the training data are discrete points, we rely on the generalization ability of the model and operations such as interpolation and extrapolation in order to generate each intended affective state anywhere in the two-dimensional space and thus the nature of the annotations is continuous.

The movements were recorded with a Vicon motion capture (mocap) system and 53 reflective markers. The final mocap data is mapped to a skeleton with 26 joints as shown in Figure 5.3. After consideration with movement experts from the Laban Institute of Movement Studies in New York, we decided to use more markers on the spine as it plays an important role in capturing the body shape changes and postures relevant to the expressive dimensions.

5.4.3 Data Processing

In order to use the data for the machine learning purposes, we change the representation of the recorded data. The raw mocap data contains a sequence of joints' rotations as well as the position and orientation of the root of the skeleton. The root defines the global position and orientation of the body within a reference coordinate system. The rest of the joints at each time frame are represented by their rotations relative to their parent joint in the skeleton hierarchy. In total, each frame is represented with 72 degrees-of-freedom (DOF).

Initially, the joint rotations were encoded using the Euler angles parameterization which defines the rotations about each axis in a local coordinate system. While widely used, Euler angles parameterization does not always guarantee correct interpolations and can result in

the loss of degrees of freedom where different combinations of each of its three components can lead to the same 3D rotation (also known as gimbal lock). Therefore, we convert the root orientation and the joints with 3 DOF to exponential maps (Grassia, 1998) in order to avoid gimbal lock and the discontinuities that occur with Euler parameterization of rotations. The final representation of the training data, after removing the dimensions that are constant, contains 52 dimensions.

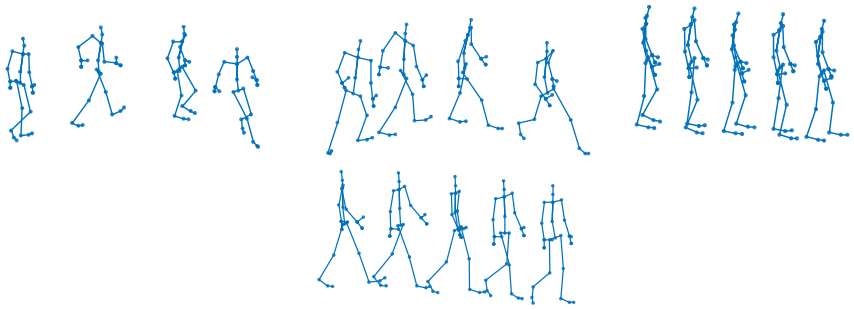


Figure 5.4: The generated walking movements. From left to right: trained on the male actor, HVHA; trained on the female actor, LVHA; trained on the male actor, LVLA; trained on the male actor, HVLA.

5.4.4 Controlling the Affect in Movement

Our model is defined by a function of the form:

$$M_t = f(M_{<t}, Fun, Exe, Exp) \tag{5.2}$$

where M_t represents the movement data at time t, $M_{<t}$ represents a set of past movement data, and the Fun, Exe, and Exp represent the function, execution, and expression dimensions of the movement, respectively. This corresponds to our goal to build a parametric motion graph which is capable of generating streams of movement that can be controlled across the three dimensions mentioned above. For example, one can specify that the agent start *walking* from a standing posture and follow a given path, while expressing a *highly aroused emotion*, and then transitioning to *sitting* on a chair with a *neutral arousal level*. In this paper, we only address the expression dimension, which is defined by:

$$Exp = (v, a) \tag{5.3}$$

where v denotes the level of valence, and a denotes the level of arousal.

In the case of computer-assisted creativity, the animator specifies his or her desired v and a values and provides some initial frames. The initial frames indicate the few poses that the movement starts from and provides a smooth continuation, where the generated movements make a transition from the emotional qualities of the initial frames to the given

emotions. In this case, the initial frames can be the last frames of the previous segment of the animation or some frames from the recorded data. By calling the function continuously for the required amount of frames, it generates movement segments that are expressive according to the v and a values.

In the case of controlling a virtual agent's movements, this function can be used as the motor control for the agent, receiving its valence and arousal values directly from the agent's affective state. At each time step, the function produces the subsequent frame of movement based on the current affective state of the agent while smoothly continuing the movement from previous time steps.

5.5 Experiments

5.5.1 Controlling the Expressivity of Movements

In order to test the model's ability to generate movements based on any given affective state of an agent, we trained an FCRBM with the movements of the actor walking a figure-8-shaped path, down-sampled to 30 frames-per-second, resulting in around 18000 frames that contain all the combinations of the valence and arousal levels of one actor. The data included all the combinations of high, neutral, and low levels of valence and arousal. We used an FCRBM with 400 hidden units, 300 factors for each three-way connection, and at each time-frame, the model was conditioned on six past frames of the data. After 600 epochs, we were able to generate good-quality new samples except for the low valence and low arousal (LVLA) movements from the female actor. We believe that this is due to the very low speed and low energy movements of the female actor for this specific combination in the training data which cannot be captured as well as other combinations by conditioning only on the past six frames. Another shortcoming of the results is the occasional foot sliding, which is due to the lack of constraints on the foot movements in the data-driven approaches.

For the generation, the context unit of the FCRBM is set to the agent's affective state while the state was fixed for each generated sequence. The model was initialized with six frames of the movement from the same affective state from the training data. The results demonstrate that the model was successful in generating new samples as shown in Figure 5.4. The videos of both the training and the generated movements can be found at `http://goo.gl/hL5kJa`.

5.5.2 Generating Transitions

The affective state of the agent can change gradually over time, and thus we are especially interested in expressing such transitions in the movements of the agent smoothly. For this experiment, we use the same model as above except that we train the model for 2000 epochs. In order to generate transitions, we gradually change the affective state of the agent, and consequently, the feature unit of the FCRBM from one point (e.g., high arousal) to another

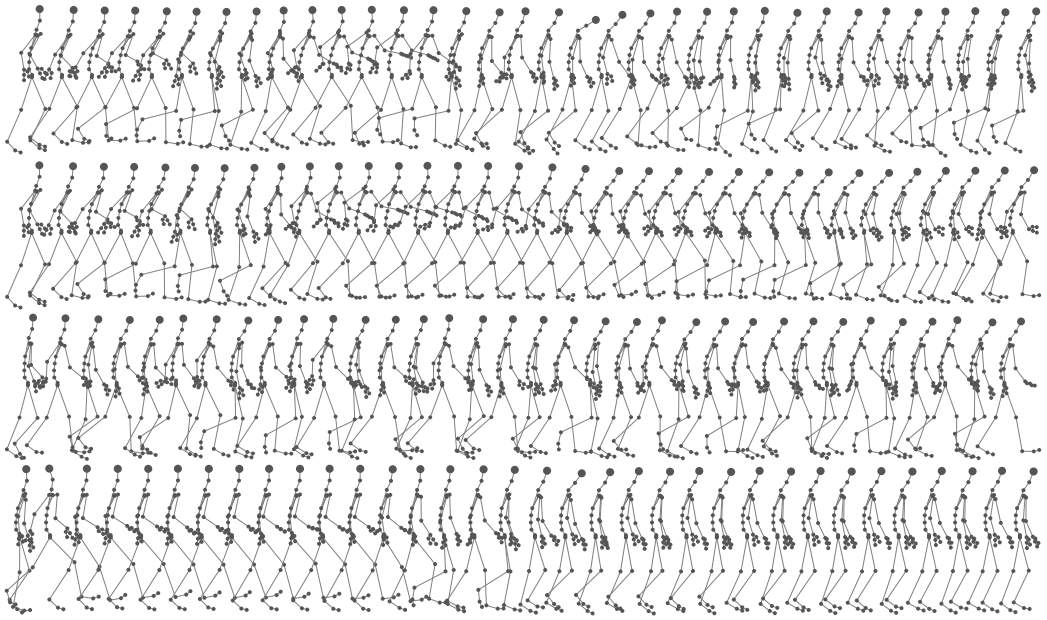


Figure 5.5: The generated transitions between two affective states. From top to bottom: HVHA to HVLA, HVHA to LVHA, HVNA to LVNA, NVHA to NVLA. Note that the figures are sampled every 16th frame and are spaced linearly for visualization purposes.

(e.g., low arousal). As shown in Figure 5.5, the generated movements smoothly reflect the changes in the state of the agent. This experiment demonstrates that the model is able to generalize the expressive characterization of movement and generate movements for the combinations of the valence and arousal that do not exist in the training data.[1]

5.5.3 Extrapolation

The model can also extrapolate slightly beyond the valence and arousal levels that were intended by the actors and exist in the training data.[2] Extrapolation can be seen as a form of exaggeration, which is suggested in the computer animation guidelines as a way to improve the perception of an affective state (Lasseter and Lasseter, 1987). For example, in order to make an agent look happy, the model should generate movements that are more happy than the intended levels by the human actors.

5.5.4 Validation of Expressivity

In order to assess the quality of the training data, as well as the ability of the system to communicate any given affective state of the agent, we conducted an experiment in which

[1]The videos of the transitions can be found at `http://goo.gl/hL5kJa`.

[2]The extrapolated movements are labeled as exaggerated and can be found at `http://goo.gl/hL5kJa`.

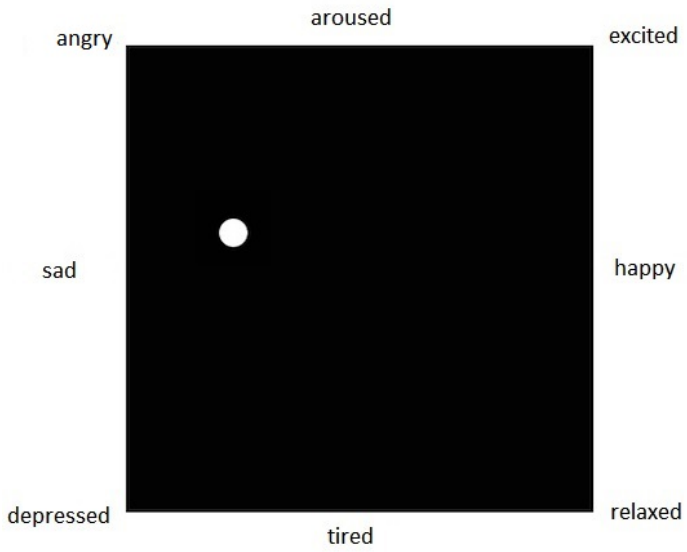


Figure 5.6: The affect grid used in the experiment to collect the participants' perception of the affective state.

human participants rated the valence and arousal levels they perceived in both generated and recorded movements.

All the movement sequences (10 generated, 12 recorded) were rendered as short video clips of simple skeletal characters. In order to focus only on bodily movements and cues, the characters did not have a face, skin, or clothing. The length of each clip was between 10 to 25 seconds. The order of the clips were randomized for each participant. The experiment was presented using a web-based questionnaire, and the participants were instructed to watch each clip and use a 2-dimensional affect grid (Figure 5.6) to rate their perceived valence and arousal levels in the movements. The ratings along each dimension were mapped to the range of $[-1, +1]$ In order to avoid any bias against the computer-generated content, the participants were not told that some clips represented computer-generated movements.

Fifteen undergraduate students, in a third-year computer animation course and gathered in a classroom, participated in the experiment. Instructions were given to them on how to use the on-line questionnaire before they individually started watching the videos. There was no time limit for the students to finish the experiment, and they could watch each clip as many times as they wanted.

The means of the responses for the valence and arousal ratings are shown in Figure 5.7, with their statistics detailed in Table 5.1. The participants could successfully classify the arousal levels as high, neutral, and low, although they perceived these levels with less intensity compared to the instructed levels. For example, the high arousal recorded movements were rated on average as 0.44 out of 1.0 in contrast to 1.0 out of 1.0. Overall, the analysis

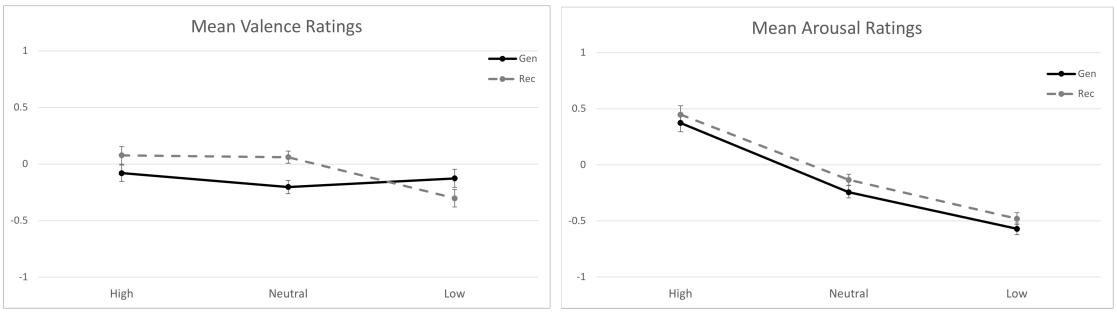


Figure 5.7: The mean ratings for valence (left) and arousal (right) for the recorded (rec) and the generated (gen) movements. The size of the error bars represents the Standard Error.

	Recorded Data	**Generated Data**
High Valence	Mean = 0.077, SD = 0.518, SE = 0.077	Mean = -0.801, SD = 0.482, SE = 0.071
Neutral Valence	Mean = 0.061, SD = 0.509, SE = 0.054	Mean = -0.203, SD = 0.454, SE = 0.059
Low Valence	Mean = -0.302, SD = 0.517, SE = 0.077	Mean = -0.126, SD = 0.542, SE = 0.081
High Arousal	Mean = 0.449, SD = 0.540, SE = 0.079	Mean = 0.374, SD = 0.408, SE = 0.060
Neutral Arousal	Mean = -0.133, SD = 0.480, SE = 0.050	Mean = -0.243, SD = 0.473, SE = 0.061
Low Arousal	Mean = -0.480, SD = 0.357, SE = 0.053	Mean = -0.570, SD = 0.350, SE = 0.052

Table 5.1: Statistics of the responses for both the recorded and the generated data (SD stands for Standard Deviation and SE stands for Standard Error.)

of the responses shows high inter-rater reliability (Cronbach's $\alpha = 0.89$ for valence, 0.98 for arousal).

The participants could identify the neutral and low valence levels correctly, while their ratings of the high valence movements averaged near the center of the spectrum. This suggests that perhaps other cues beyond bodily movements, such as facial expression or voice, are necessary to correctly express and identify the valence level. Another possibility is that the performance of the actors did not contain enough variations along the valence dimension.

The perceptions of the participants for arousal and valence levels of the recorded and generated movements are compared using the Mann-Whitney U test. The arousal level was perceived the same between the recorded and generated movements ($U = 14541, p < 0.227, N1 = 150, N2 = 180$). However, the valence level of the generated data was marginally perceived as less than the recorded data ($U = 15294, p < 0.038, N1 = 150, N2 = 180$). Overall, the mean rating of the valence of the generated movements is slightly lower than the recorded movements.

5.6 Conclusions

We presented a generative model of affect-expressive movements, which allows controlling the emotional qualities of its output. The emotional qualities are represented and modulated by two continuous variables describing the valence and the arousal level of the agent. We applied the model on a data set of walking movements performed by two professional actors while modulating their movements based on different combinations of valence and arousal levels. The validation results show that the model can successfully express the affect along the arousal dimension. However, expressing the valence is shown to be not sufficient at the moment.

As future work, we plan to extend our model towards the following directions: (1) improve the expression of the valence dimension; (2) create a model that allows controlling the function and execution dimensions of the movement as well as its expression, extending the previous works on parametric motion graphs to support affect expression; (3) use the ground-truth labels from the experiment to train the model; and (4) use more dimension than valence and arousal to represent affect, as suggested in the literature (Fontaine et al., 2007).

Bibliography

J. Bates. 1994. The role of emotion in believable agents. *Communications of the ACM* 37, 7 (1994), 122–125.

M. Brand and A. Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 183–192.

D. Chi, M. Costa, L. Zhao, and N. Badler. 2000. The EMOTE model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press, 173–182.

C. Chiu and S. Marsella. 2011. A style controller for generating virtual human behaviors. In *The th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1023–1030.

J. R. J. Fontaine, K. R. Scherer, E. B. Roesch, and P. C. Ellsworth. 2007. The World of Emotions is not Two-Dimensional. *Psychological Science* 18, 12 (Dec. 2007), 1050–1057.

F. S. Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48.

J. Huang and C. Pelachaud. 2012. Expressive Body Animation Pipeline for Virtual Agent. In *Intelligent Virtual Agents*, Yukiko Nakano, Michael Neff, Ana Paiva, and Marilyn Walker (Eds.). Springer, 355–362.

M. Karg, A.-A. Samadani, R. Gorbet, K. Kuhnlenz, J. Hoey, and D. Kulic. 2013. Body Movements for Affective Expression: A Survey of Automatic Recognition and Generation. *IEEE Transactions on Affective Computing* 4, 4 (2013), 341–359.

L. Kovar, M. Gleicher, and F. Pighin. 2002. Motion Graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*. 473–482.

T. Kwon and S. Y. Shin. 2005. Motion modeling for on-line locomotion synthesis. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 29–38.

J. Lasseter and J. Lasseter. 1987. *Principles of traditional animation applied to 3D computer animation*. Vol. 21. ACM.

Y. Ma, H. M. Paterson, and F. E. Pollick. 2006. A motion capture library for the study of identity, gender, and emotion perception from biological motion. *Behavior Research Methods* 38, 1 (2006), 134–141.

A. Mehrabian. 1996. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament. *Current Psychology* 14, 4 (Dec. 1996), 261–292.

N. Niwase, J. Yamagishi, and T. Kobayashi. 2005. Human Walking Motion Synthesis with Desired Pace and Stride Length Based on HSMM. *IEICE TRANSACTIONS on Information and Systems* E88-D, 11 (Nov. 2005), 2492–2499.

R. Plutchik and H. R. Conte. 1997. *Circumplex Models of Personality and Emotions*. American Psychological Association.

J. A. Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology* 39, 6 (Dec. 1980), 1161–1178.

A. Shoulson, N. Marshak, M. Kapadia, and N. I. Badler. 2013. ADAPT: the agent development and prototyping testbed. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 9–18.

L. M. Tanco and A. Hilton. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Workshop on Human Motion*. IEEE, 137–142.

G. W. Taylor. 2009. *Composable, distributed-state models for high-dimensional time series.* Ph.D. Dissertation. University of Toronto.

G. W. Taylor and G. E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, 1025–1032.

G. W. Taylor, G. E. Hinton, and S. Roweis. 2006. Modeling Human Motion Using Binary Latent Variables. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06)*. MIT Press, 1345–1352.

J. Tilmanne and T. Dutoit. 2010. Expressive gait synthesis using PCA and Gaussian modeling. In *Proceedings of the Third international conference on Motion in games*. Springer-Verlag, 363–374.

J. L. Tracy and D. Randles. 2011. Four Models of Basic Emotions: A Review of Ekman and Cordaro, Izard, Levenson, and Panksepp and Watt. *Emotion Review* 3, 4 (Oct. 2011), 397–405.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 975–982.

Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. 2006. Learning Style-directed Dynamics of Human Motion for Automatic Motion Synthesis. In *IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06*. 4428–4433.

M. Wooldridge. 2009. *An Introduction to MultiAgent Systems.* John Wiley & Sons.

Chapter 6

WalkNet: Interactive Walking Movement Controller

This chapter presents content from the following paper:

Omid Alemi and Philippe Pasquier. 2017. WalkNet: A Neural-Network-Based Interactive Walking Controller. In Proceedings of the 17th International Conference on Intelligent Virtual Agents (Lecture Notes in Computer Science), Vol. 10498. Springer, 15–24. DOI: https://doi.org/10.1007/978-3-319-67401-8_2

Abstract

We present WalkNet, an interactive agent walking movement controller based on neural networks. WalkNet supports controlling the agent's walking movements with high-level factors that are semantically meaningful, providing an interface between the agent and its movements in such a way that the characteristics of the movements can be directly determined by the internal state of the agent. The controlling factors are defined across the dimensions of planning, affect expression, and personal movement signature. WalkNet employs Factored, Conditional Restricted Boltzmann Machines to learn and generate movements. We train the model on a corpus of motion capture data that contains movements from multiple human subjects, multiple affect expressions, and multiple walking trajectories. The generation process is real-time and is not memory intensive. WalkNet can be used both in interactive scenarios in which it is controlled by a human user and in scenarios in which it is driven by another AI component.

6.1 Introduction

Data-driven movement animation manipulation and generation techniques use recorded motion capture data to preserve the realism of their output while providing some level of control and manipulation. This makes them more suitable for generating affect-expressive movements, compared to the physics-based approaches to modelling and generating movement animation. Data-driven techniques bring the possibility of augmenting a corpus of motion capture data so that human animators have more assets at their disposal. Furthermore, one can use movement generation models in interactive scenarios, where a human user or an algorithm controls the behaviour of the animated agent in real-time.

With the increasing demand for content for nonlinear media such as video games, a movement controller that supports generating movements in real-time based on the given descriptions has applications in AI-based agent animation, interactive agent control, as well as crowd simulation.

Data-driven methods allow for manipulation of the motion capture data, either by concatenating, blending, or learning and then generating data. Concatenation methods repeat and reuse the movements in a motion capture corpus by rearranging them, making longer streams of movements from shorter segments. In blending, the representations of two or more motion capture segments are combined to create a new segment that exhibits characteristics from the blended segments. Compared to other techniques, machine learning models are better at generalizing over the variations in the data and generating movements that do not exist in their training corpus. Some of the machine learning techniques also provide mechanisms for controlling and manipulating what is being generated, making them suitable for controlling virtual agents.

The body of the research on machine-learning-based movement generation has some challenges. Controlling the movements of an agent requires a description of the movement to be generated, and a machine learning model that is capable of mapping those descriptions to movement, in real-time. In this regard, the majority of the works suffer from one or more of the following: (1) they do not support controlling the generated movements (e.g., (Crnkovic-Friis and Crnkovic-Friis, 2016)), (2) they only support controlling a single factor (e.g., (Alemi et al., 2015)), (3) the controlling factor is often not clearly defined with respect to an agent's internal state(e.g., (Tilmanne et al., 2014)), or (4) the generation process is computationally and/or memory intensive (e.g., (Wang et al., 2007)).

In order to overcome the above limitations, we present WalkNet, a walking movement controller for animated virtual agents. At its core, WalkNet uses a neural network to learn and generate its movements based on a set of given controlling factors. The factors are chosen to work directly with the internal state of the agent, corresponding to the planning, expression, and personal movement signature dimensions of movement. In the future, we intend to extend the model to support controlling the functional dimension as well. The

agent can plan its walking movements based on any given trajectory. The affective state is modelled by the valence and arousal dimensions of affect. Furthermore, the movement generation model is capable of exhibiting distinctive personal movement signatures (styles). This allows for using the same model for a group of agents that each portrays a different character. The main contributions of our approach are summarized below:

- WalkNet provides control over multiple dimensions of movement in a single model.
- Learned over a limited sample of affective states (i.e., only high, neutral, and low points) and only two human subjects, WalkNet learns a generalized space of affect and movement signature.
- The generation process is real-time. Unlike graph and tree-based structures, there is no need for search or optimization to generate desired movements.

6.2 Background and Related Work

Controlling Movement Generation In data-driven movement-generation approaches, different techniques, and the combinations of them are used to control and manipulate the characteristics and qualities of motion capture data. These include organizing the data using specialized data structures, such as motion graphs (Kovar et al., 2002; Heck and Gleicher, 2007), as well as blending and interpolating multiple segments. Regarding the machine learning models, there are multiple ways that they support controlling the generation: 1) Train a separate model for each point in the factor space. Each model is trained only on the data that corresponds to that particular point, thus only imitating the same factor value. To control the generation, one has to switch between the models. 2) Using a parametric probability distribution, in which the parameters of the distribution are a function of the controlling factors (Herzog et al., 2008), allows for controlling the statistical characteristics of the generated data. 3) By designing the machine learning model in a way that provides a mechanism for a factor variable to control the characteristics of the generated movements. In particular, Factored Conditional Restricted Boltzmann Machine (FCRBM) uses a context variable (Figure 6.3.b) that controls the behaviour of the network through gated connection between the observations and the hidden variables (Taylor and Hinton, 2009).

Machine Learning Methods for Movement Generation Machine learning models that are used for learning and generating motion capture data range from dimensionality reduction (DR) techniques (e.g., (Samadani et al., 2013)), to the Gaussian Process Latent Variable Models (GPLVMs) (e.g., (Wang et al., 2007)), Hidden Markov Models (HMMs) (e.g., (Brand and Hertzmann, 2000)), temporal variations of the Restricted Boltzmann Machines (e.g., (Taylor and Hinton, 2009; Alemi et al., 2015)), Recurrent Neural Networks (e.g, (Crnkovic-Friis and Crnkovic-Friis, 2016)), and Convolutional Autoencoders combined with Feed-Forward Networks (e.g., (Holden et al., 2016)).

DR techniques do not handle the temporality of the motion capture data. Furthermore, the dimensionality-reduction-based techniques rely on preprocessing steps such as sequence alignments and fixed-length representation of the data. The main limitation of the GPLVMs is that they demand heavy computational and memory resources, which makes them unsuitable for real-time generation. HMMs overcome the limitations of the two aforementioned families of models but provide a limited expressive power regarding capturing the variations in the data. Neural networks provide a better expressive power than HMMs. Convolutional Autoencoders have shown promising results in generating motion capture data and offline controlling (Holden et al., 2016). Factored Conditional RBM (FCRBM), with its special architecture that is designed to support controlling the properties of the generated data, has shown to be able to generate movements in real-time, and learn a generalized space of the movement variations (Taylor and Hinton, 2009; Alemi et al., 2015).

Affect-Expressive Movement Generation Taubert et al. (2011) combine a Gaussian process latent variable model (GP-LVM) with a standard HMM that learns the dynamics of the handshakes, encoded by the emotion information. Samadani et al. (2013) use functional principal component analysis (FPCA) to generate hand movements. Alemi et al. (2015) train an FCRBM to control the valence and arousal dimensions of walking movements.

Our Approach We build WalkNet on top of the previous work by the same authors (Alemi et al., 2015), extending the affect-expression control with the walking planning and personal movement signature. Our work differs from graph-like structures as it does not require to build an explicit and fixed data-structure, does not require search and optimization for generating movement, and does not require storing the movement data for generation. It also differs from the work of Crnkovic-Friis and Crnkovic-Friis (2016) as it provides a mechanism to control the generated data. It allows for real-time and iterative generation compared to the work of Holden et al. (2016).

6.3 Training Data

For training the model, we use a set of motion capture data that provides movements with variations in walking direction (planning), the valence and arousal levels (expression), and the personal movement signature. As we could not find a publicly available motion capture database that provides movements with such variations, we recorded our own set of training data. The complete data set is publicly accessible in the MoDa database[1].

The training data includes the movements of two professional actors and dancers (one female, one male). Each subject walks following a curved figure-8-shaped path. The turning variations in this pattern allow the machine learning model to learn a generalized space of turning directions. To capture a space of affect-expression, each subject performs each move-

[1] `http://moda.movingstories.ca/projects/29-affective-motion-graph`

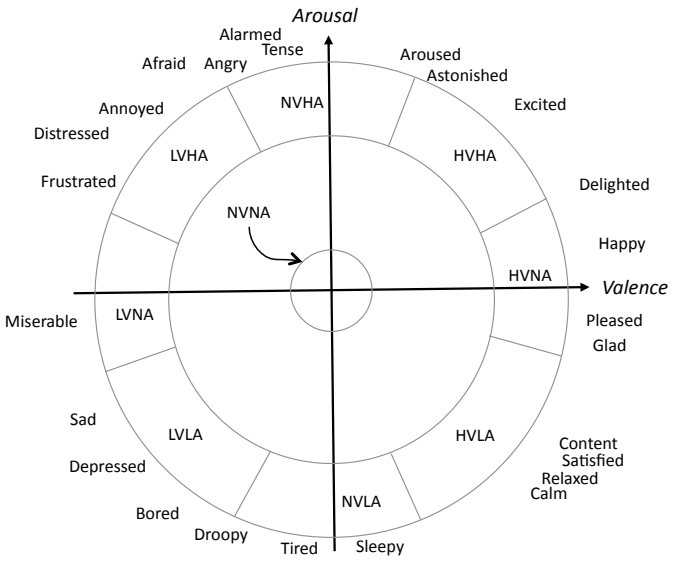


Figure 6.1: The affect model described by valence and arousal dimensions with the 9 zones recorded in the training data. The mapping to the categorical emotion labels are based on (Plutchik and Conte, 1997). H: high, N: neutral, L: low, V: valence, and A: arousal.

ment with nine different expressions along the valence and arousal dimensions (Plutchik and Conte, 1997), shown in Figure 6.1. Using the dimensional representation of affect over the categorical systems allows for interpolation and extrapolation of the affect states, as well as transitions. Each valence and arousal combination is repeated four times to capture enough motor variabilities.

The original motion capture data consists of a skeleton with 30 joints, resulting in 93 dimensions including the root position, with their rotations represented in Euler angles. The data is captured at 120 frames-per-second. We use exponential maps (Grassia, 1998) to represent joint angles to avoid loss of degrees-of-freedom and discontinuities. We replace the skeleton root orientation and translation by the delta values of the translational velocity of the root along the floor plane, as well as its rotational velocity along the axis perpendicular to the floor plane. We remove the dimensions of the data that are constant or zero and downsample the data to 30 frames-per-seconds. The final data set used for the training consist of 18 motion capture segments (2 subjects $\times$ 9 affective states), containing 37,562 frames in total, with 52-dimensional frames.

6.4 The Walking Controller

System Overview As shown in Figure 6.2, at the core of the WalkNet, the movement generator, a Factored, Conditional Restricted Boltzmann Machine (FCRBM), generates a continuous stream of movement. The movement stream is modulated by a set of controlling

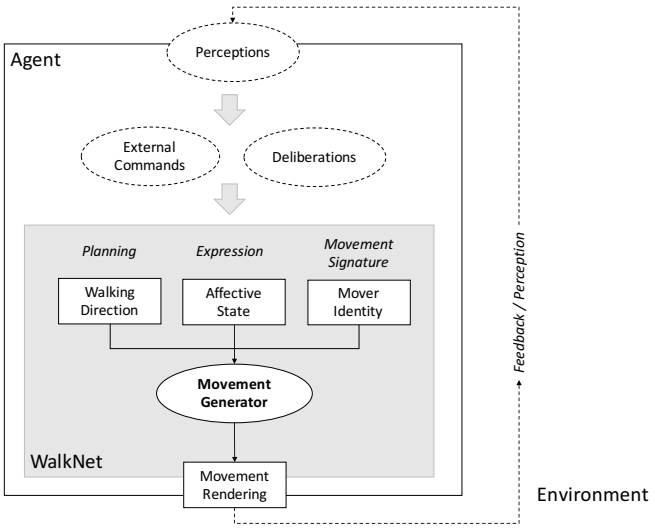


Figure 6.2: The WalkNet controller, embedded in an agent model.

factors, determined from the internal state of the agent or through external commands. From an agency perspective, we organize these factors into different dimensions, mainly the *function*, the *planning*, the *expression*, and factors that together make the *personal movement signature* of the agent. WalkNet does not make any assumptions on how the agent movement descriptor is set. Thus, making it flexible to be integrated into various agent models for different applications.

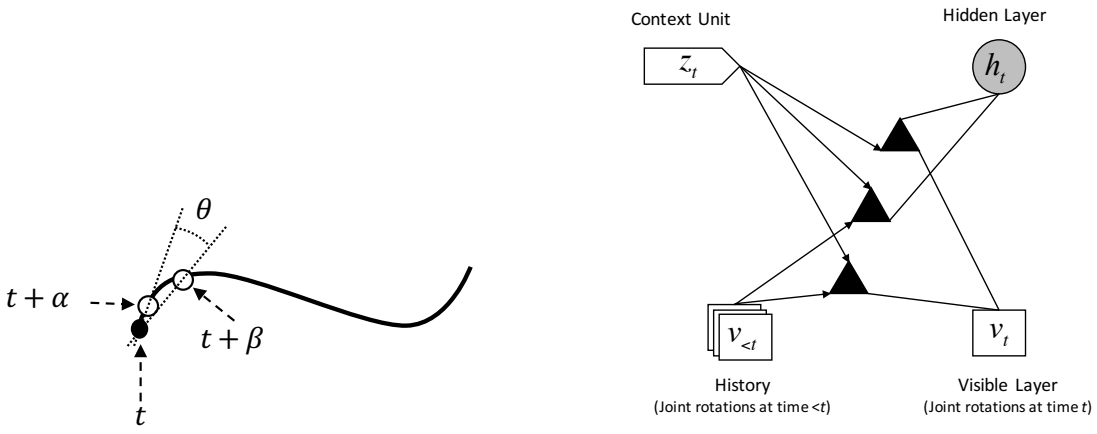
Agent Movement Descriptor We use an agent movement descriptor AMD to formalize the contributing factors to the agent's movements at time t along the dimensions of function (F), planning (P), expression (E), and personal movement signature (S):

$$\text{AMD}_t = \langle F_t, P_t, E_t, S_t \rangle$$

In WalkNet, the F is always set to walking. The planning dimension of walking is defined by $P_t = \langle D_t \rangle$ where D_t represents the direction that the agent intends to walk towards, relative to its current orientation. The expression dimension is defined by $E_t = \langle V_t, A_t \rangle$ where V_t and A_t stand for valance and arousal levels at time t respectively. Currently, we use the actor/performer's identity as a proxy to model the personal movement signature, through a weighted combination of a K-dimensional vector, representing K subjects:

$$S_t = \{I_t^1, I_t^2, .., I_t^K | \sum_k^K I_t^k = 1\}$$

We recognize that this is a simple way of capturing movement signature. In the future, we plan on learning a representation that captures the personal movement signature.



(a) (b)

Figure 6.3: (a): Calculating the direction of the subject in the training data. (b): FCRBM's architecture with valence and arousal labels modulating the interactions between the past visible, current visible, and current hidden units.

Training Data Annotation Here, we describe how we annotate our data to capture different states of the factors in the agent movement descriptor.

As we have two human subjects in our training data, we use a 2-dimensional label with a one-hot encoding scheme for the movement signature.

We use the valence and arousal representation of affect to annotate the expression of affect. Each movement segment in the training data is labeled with low, neutral, and high for both their valence and arousal levels. After experimenting with different ranges, we use the values of 1, 2, and 3 to represent low, neutral, and high levels in the annotations. Although the training labels are discrete, the valence and arousal values are continuous in nature, and for the generation, one can specify any real value within the range of $[0, 4]$, as the FCRBM is able to interpolate or extrapolate between those discrete states.

For annotating the heading direction, we determine the labels using a method that is inspired from Kovar et al. (2002). For the label at frame t, considering the projection of the traveled path of the skeleton root on the ground floor, we select two points on the path, one at a very close distance to the current location, and another one at a slightly further location from the current location (Figure 6.3.a). We calculate the angle between the two lines that result from connecting the two chosen points and use this angle as a measure of the heading direction. After scaling the angle to have a value between -1 and +1, directions towards the right of the subject are associated with positive numbers, and directions towards the left of the subject are associated with negative numbers.

Initial experiments showed that using only a one-dimensional label vector for modelling the direction parameter causes poor results when asking the model to generate movement for the values that are around the center of the continuum. The problem arises from the

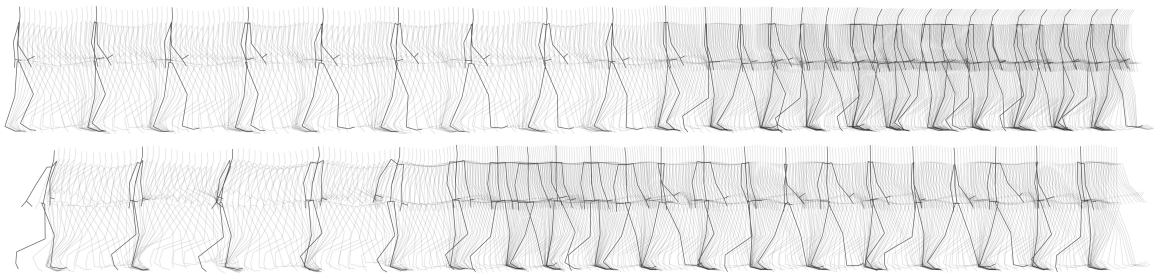


Figure 6.4: WalkNet's output. Top: making a transition from a high valence and high arousal affective state to a low valence and low arousal state. Bottom: making a transition from a low valence and high arousal affective state to a high valence and low arousal state.

fact that the model associates high values with one end of the spectrum and low values with the other end of the spectrum, while semantically, there is no difference between each end. This issue is overcome by using a two-dimensional vector $D = [L, R]$ to annotate the two polarities of the direction. The two dimensions of this vector complement each other, following the relationship $R = 1 - L$ in a normalized case. Therefore, the direction is encoded as two labels, one for right and one for left.

As a result, each frame t of the training segment s is annotated with a 6-dimensional label of the form:

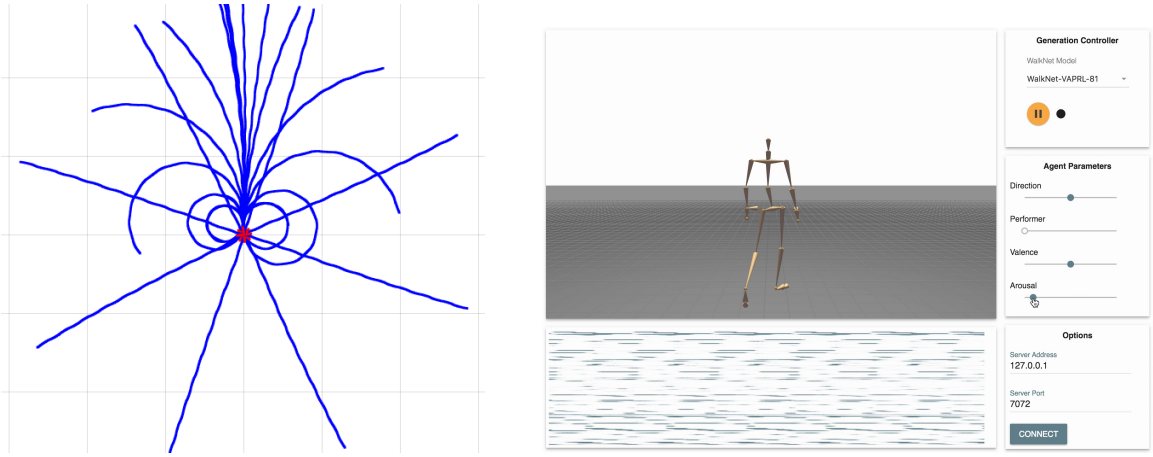
$$L_t^s = \langle I^{s^1}, I^{s^2}, V^s, A^s, R_t, L_t \rangle$$

Note that as the identity of the subject and the valence and arousal levels are fixed for each training segment, only R_t and L_t values are changed between each frame of the same segment.

Movement Generator We use an FCRBM to generate the movements of the agent. As shown in Figure 6.3.b, FCRBM learns the autoregressive, as well as the nonlinear temporal patterns in a time-series. Every weight in FCRBM is modulated by the value of its context unit, making it possible to change the energy landscape of the model by changing the value of the context unit, and effectively controlling the model's prediction. In WalkNet, the FCRBM learns to predict the next motion capture frame, given a recent history of the motion capture frames, as well as the movement descriptors fed to its context unit Z. This results in a predictive function in the form of:

$$M_t = f(M_{<t}, Z_t), Z_t = \langle I_t^1, I_t^2, V_t, A_t, R_t, L_t \rangle$$

By iteratively calling this function and feeding it with the generated frames from the previous cycles, we can continuously generate movements that are modulated by the given descriptors.



(a) (b)

Figure 6.5: (a) The projection of the agent's movements on the ground floor plane, making turns with different angles. (b) The interactive controller.

6.5 Results

In this section, we demonstrate the capabilities of WalkNet in generating realistic motion capture data. We use an FCRBM with 150 hidden units and 400 factors, trained for 3000 epochs. The model takes 12 past motion capture frames as input and predicts the next frame, modulated by a vector of 6 dimensions (Z).

Affect Expression By specifying different values for the valence and arousal levels in the agent movement descriptors, WalkNet can generate a variety of affect expressions, even for those values that do not exist in the training data. This allows for generating walking movements for any point in a range of $[0, 4]$. With this, one can not only generate walking movements for high, neutral, and low levels of valence and arousal but also make transitions from one state to another (Figure 6.4).

In a previous work by the same authors (Alemi et al., 2015), a study was conducted to validate the expressiveness of the movements. The analysis shows that independent human observers can successfully identify different levels of arousal. However, they can only correctly identify the low valence levels, and often confused the neural and high valence levels. The analysis reached the same results for the recorded movements of human actors as well. We believe that due to the lack of facial expression, recognition of valence through movements, as represented by a stick figure, is often challenging for humans.

Movement Signature WalkNet can generate signatures that are interpolations between the two actors. The difference in the generated movement signatures are demonstrated in the accompanied video[2].

[2] `https://youtu.be/3JBfGF4tsmA`

Navigation As the results are demonstrated in Figure 6.5.a, different values for the direction factor generates movements along curves with different curvatures.

Interactive Control WalkNet through a graphical user interface (GUI) developed for this purpose. The GUI allows the user to choose the parameters of the model, while the agent's movements are rendered in 3D in real-time. A snapshot of the GUI is shown in Figure 6.5.b. A video of the GUI is also provided[2].

Generating each frame takes 0.0063 seconds on a MacBook Pro with an Intel(R) Core(TM) i7-4850HQ CPU at 2.30GHz. Thus, at 30 frames-per-second, it takes 0.1890 of a second to generate the movements for each second.

6.6 Conclusion and Future Work

This paper introduces WalkNet, a walking movement controller. It can generate realistically-looking walking movements in real-time, while modulating them using an agent movement descriptor that specifies the expression of affect through the movement, the walking direction, and the personal movement signature of the agent.

WalkNet is designed with integration into agent models in mind. It does not make any assumption on how the movement descriptor is specified, making it possible to be used in interactive scenarios, in which a user directly controls the agent's movements, or in scripted or AI-driven applications. For example, given a target path to follow, by observing the traveled path, the agent can continuously correct its course to stay on the target path.

In future, we plan to perform more formal and quantitative evaluation of the model. Furthermore, we intend to use more human subjects in the training data. Another future direction is to extend the model to include more than one type of movement (function). For example, allowing the agent to switch from walking to standing to sitting while performing hand gestures.

6.7 Acknowledgments

This work is funded by the Social Sciences and Humanities Research Council of Canada (SSHRC) through the Moving Stories Project, as well as the Natural Sciences and Engineering Research Council of Canada (NSERC).

Bibliography

O. Alemi, W. Li, and P. Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*. 442–448.

M. Brand and A. Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 183–192.

L. Crnkovic-Friis and L. Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. *CoRR* abs/1605.06921 (May 2016).

F. S. Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48.

R. Heck and M. Gleicher. 2007. Parametric motion graphs. In *Proceedings of the 29th Representation Learning Workshop. International Conference on Machine Learning*. ACM, 129–136.

D. Herzog, V. Krueger, and D. Grest. 2008. Parametric Hidden Markov Models for Recognition and Synthesis of Movements. In *Proceedings of the British Machine Vision Conference*. 163–172.

D. Holden, J. Saito, and T. Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics* 35, 4, Article 138 (July 2016), 11 pages.

L. Kovar, M. Gleicher, and F. Pighin. 2002. Motion Graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*. 473–482.

R. Plutchik and H. R. Conte. 1997. *Circumplex Models of Personality and Emotions*. American Psychological Association.

A.-A. Samadani, E. Kubica, R. Gorbet, and D. Kulić. 2013. Perception and Generation of Affective Hand Movements. *International Journal of Social Robotics* 5, 1 (2013), 35–51.

N. Taubert, D. Endres, A. Christensen, and M. A. Giese. 2011. Shaking Hands in Latent Space - Modeling Emotional Interactions with Gaussian Process Latent Variable Models. In *KI 2011: Advances in Artificial Intelligence*. Springer, 330–334.

G. W. Taylor and G. E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, 1025–1032.

J. Tilmanne, N. dAlessandro, M. Astrinaki, and T. Ravet. 2014. Exploration of a Stylistic Motion Space Through Realtime Synthesis. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications*. 803–809.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 975–982.

Chapter 7

GrooveNet 1.0: Music-Driven Dance Generation - Preliminary Results

This chapter presents content from the following paper:

Omid Alemi, Jules Françoise, and Philippe Pasquier. 2017. GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks. Poster accepted to the Work- shop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining - https://ml4creativity.mybluemix.net/.

Abstract

We present the preliminary results of GrooveNet, a generative system that learns to synthesize dance movements for a given audio track in real-time. Our intended application for GrooveNet is a public interactive installation in which the audience can provide their own music to interact with an avatar. We investigate training artificial neural networks, in particular Factored Conditional Restricted Boltzmann Machines (FCRBM) and Recurrent Neural Networks (RNN), on a small dataset of synchronized music and motion capture recordings of dance movements. We have captured a dataset of four dance performances for the purpose of this project. Our initial results show that we can train the FCRBM on this small dataset to generate dance movements. However, the model cannot generalize well to music tracks beyond the training data. We outline our plans to further develop GrooveNet.
Supplementary Material:`https://omid.al/groovenet-material-ml4c`

7.1 Introduction

Generating human movement remains one of the most challenging problems in computational modelling: movement is continuous, highly dimensional, and fundamentally expressive. Recognizing and generating everyday movements such as walking demands the development of elaborate models that can capture the coordination of a large set of joints (Tilmanne and Dutoit, 2010; Alemi et al., 2015).

In this paper, we address the problem of movement generation for the case of dance, a creative activity that best illustrates the complexity and expressiveness of human movement. Dancing involves complex cognitive and sensorimotor processes: it requires both fine motor control and equilibrium, accurate timing and rhythmic synchronization, memory and imagery, as well as aesthetic qualities (Bläsing et al., 2012). The way we dance in response to music depends on the genre of both dance and music, the expertise of the dancer, and their interpretation of the music in real-time. Beyond mere synchronization processes, there is no well-established relationship between movements features and musical features, except in simplified cases such as sound tracing (Nymoen et al., 2011; Caramiaux et al., 2011). As a result, generating dance movements from music is a highly non-linear and time-dependent mapping problem.

In this paper, we investigate how machine learning can capture the cross-modal dependencies between synchronized sequences of musical features and movement parameters. We present *GrooveNet*, a system for real-time music-driven dance generation that uses artificial neural networks to learn the relationships between audio features and motion capture data.

The generation of human-like creative movement is necessary in a wide range of applications, spanning animation, gaming, virtual reality, and virtual characters. Our primary field of application is artistic, and aims to explore the possibilities of computer-generated movement for creative purposes. Our music-to-dance generative system will be used in a public interactive installation allowing the audience to affect the movements of a dancing avatar by playing their own music. The avatar will be rendered with non-realistic visualizations of human movement through a holographic display.

GrooveNet relies on a machine learning model trained on a set of recordings of dance movements performed with dance music. We formulate this problem as learning the effects of one sequence on another sequence, in which both sequences are defined along a relatively dense time dimension (e.g., as opposed to text). The model is trained from synchronized sequences of dance and music features, in order to generate new movements from a new music track.

The development of GrooveNet faced a number of challenges related to the task of learning how dance movements are coordinated with music. First, as we already discussed, the mapping between audio features and movement parameters is highly non-linear and is require the model to learn and embed complex temporal structures. Second, the model

should learn efficiently from a relatively small dataset. There are no publicly available dataset that contains synchronized pairs of dance and music in the form of motion capture data and raw audio features. We recorded a dataset of 3 dance performances, raising about 23 minutes of synchronized movement and audio data. Third, our main application consists in a public interactive installation allowing audience member to provide their own music to a dancing avatar. This demands that movements are generated in real-time from an audio stream, and that the algorithm generalizes to new, and possibly unheard, music sequences.

7.2 Related Work

In this section, we review movement generation techniques that rely on machine learning with a particular focus on dance movements and audio-driven approaches.

7.2.1 Machine-Learning-Based Movement Generation

A variety of machine learning models are used for learning and generating human movement in the form of motion capture data. They range from dimensionality reduction techniques (Tilmanne and Dutoit, 2010; Samadani et al., 2013), to Hidden Markov Models (Brand and Hertzmann, 2000), Gaussian Processes (Wang et al., 2007), and neural networks (Taylor and Hinton, 2009; Crnkovic-Friis and Crnkovic-Friis, 2016; Holden et al., 2016).

Dimensionality reduction techniques can capture the underlying correlations behind the joint rotations representing the postures in motion capture data (Tilmanne and Dutoit, 2010; Samadani et al., 2013). However, such techniques require pre-processing steps such as sequence alignments and fixed-length representation of the data, which limit their application to real-world dance data. Most importantly, their inability to handle the temporality of the movement data is critically limiting for movement generation. Gaussian Process Latent Variable Models (GPLVMs) (Wang et al., 2007) can efficiently generalize over the variations in human movement, but they are limited by heavy computational and memory resources, which makes them unsuitable for real-time generation. Hidden Markov Models (HMMs) overcome the limitations of the two aforementioned family of models (Brand and Hertzmann, 2000), but provide a limited expressive power in terms of capturing the variations in the data.

Neural networks provide a better expressive power than HMMs. Convolutional Autoencoders have shown promising results in generating motion capture data and offline controlling (Holden et al., 2016). Factored Conditional Restricted Boltzmann Machines (FCRBMs), with their special architecture that is designed to support controlling the properties of the generated data, have shown to be able to generate movements in real-time, and learn a generalized space of the movement variations (Taylor and Hinton, 2009; Alemi et al., 2015). In addition, Recurrent Neural Networks (RNNs), and in particular Long Short-Term Memory

RNNs (LSTM-RNNs) are used to learning and generation movements (Crnkovic-Friis and Crnkovic-Friis, 2016) in an unsupervised and uncontrolled manner.

7.2.2 Dance Movement Generation

Many of the existing machine learning-based movement generation techniques have been applied to dance. Hidden Markov Models and their extensions have been applied to the synthesis of dance movements (Brand and Hertzmann, 2000; Ofli et al., 2012). In particular, Wang et al. trained Hierarchical Hidden Markov Models with non-parametric output distributions (NPHHMM) on motion capture data containing ballet walk, ballet roll, disco, and complex disco (Wang et al., 2005). Other approaches rely on dynamical systems modelling to capture the dynamics of dance movements. Li et al. (2002) used Linear Dynamical Systems (LDS) to learn and generate dance movements. They train their model on 20 minutes of dance motion of a professional dancer, performing mostly disco. Their model automatically learns motion textons, representing local movement dynamics. The intuition behind the textons is that each complex movement sequence consists of simple repetitive patterns. For example, a dance sequence might consist of repeated moves such as spin, hop, kick, and tiptoeing. The approach allows for real-time synthesis and provides a number of ways to generate movements, such as key-framing and noise-driven generation.

Recently, artificial neural networks have been successfully applied to the synthesis of dance movements. Donahue et al. (2017) focused on generating choreographies, represented as step charts that encode the timing and position of steps, for the Dance Dance Revolution game. They used LSTMs to generate a new step chart, given a raw audio track. Their method, however, is limited to the generation of sequences of discrete step indicators rather than continuous movements. Crnkovic-Friis and Crnkovic-Friis (2016) used Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) to learn and generate choreography. They trained the model on 6 hours of contemporary dance data captured using Microsoft Kinect. This approach does not provide any methods for controlling the generation and does not accompany music.

Controlling ML-based Movement Generation

There are a number of different methods to control the qualities of the movements generated by a machine learning model: 1) train a separate model for each realization of a movement quality, 2) use parametric statistical distributions to capture the variations of movement (Herzog et al., 2008; Yamazaki et al., 2005), and 3) design machine learning models specifically to accommodate the task of controlling the generation process (Taylor and Hinton, 2009).

7.2.3 Audio-Driven Movement Generation

Speech-Driven Synthesis

Many approaches to movement generation for virtual avatars rely on audio signals to guarantee that the synthesized gestures are consistent with other modalities. In general, the input audio is a speech signal that drives the generation of movements of the lips (Yamamoto et al., 1998), eyebrows (Ding et al., 2013), head (Hofer, 2009), or hands (Levine et al., 2010). In this case, the goal is to ensure that the motor behavior is realistic and consistent with both the content and the expression of the input speech utterances.

Most approaches rely on probabilistic models such as Hidden Markov Models (HMMs) and extensions (Yamamoto et al., 1998; Hofer, 2009; Ding et al., 2013) or Hidden Conditional Random Fields (Levine et al., 2010). Chiu and Marsella (2011) proposed Hierarchical Factored Conditional Restricted Boltzmann Machines (HFCRBM) to learn and generate gestures, controlled by the prosody of speech. Using a set of training data that includes motion capture recordings of gestures accompanied with the voice recordings of the actors, the model learns the relationship between the prosody of speech and the movement. The model then generates novel gestures given a new set of voices. The audio features used are pitch, intensity and correlation.

Music-Driven Dance Generation

While music-driven dance generation also considers cross-modal sequence-to-sequence mapping, it is important to underline the complexity of music-to-dance mapping. While in speech the acoustic and motion signals are often generated by the same underlying process, the relationships between music and movement in dance are far more complex and arbitrary. They depend on the genre and context, the expertise and personal characteristics of the performer, and they present a complex hierarchy of temporal structures, spanning from the short-term synchronization of gestures to the beat to long-term evolutions of the dance patterns.

Ofli et al. introduced an audio-driven dancing avatar using HMM-based motion synthesis (Ofli et al., 2008). For training, their approach requires movement to be manually annotated into specific patterns (or dance figures) synchronized with the beats. For generation, the audio is segmented using beat detection and the recognition of the patterns from Mel-Frequency Cepstral Coefficients is used to select the motion patterns to generate. Their approach was further extended to include unsupervised analysis of the dance patterns (Ofli et al., 2012). Ofli et al. describe three types of models: *musical measure models* and *exchangeable figures models*, which respectively represent many-to-one and one-to-many associations between musical patterns and dance figures, as well as *figure transition model* that capture the intrinsic dependencies of dance figures. Yet, one of the main limitation of these approaches lies in the synthesis approach, that relies on the classification of the in-

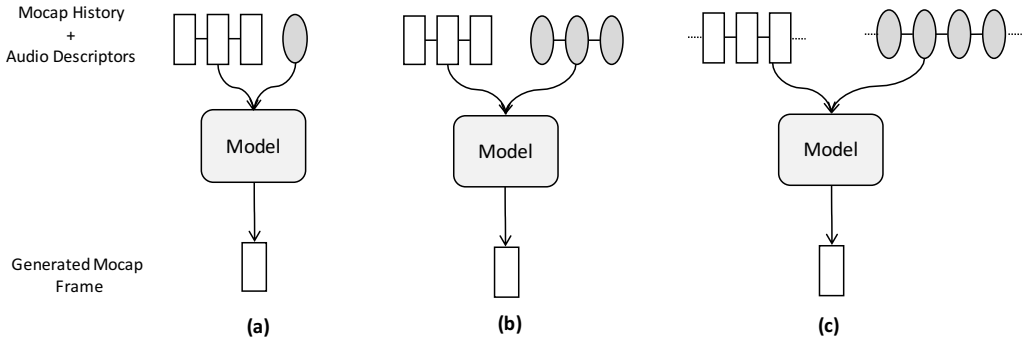


Figure 7.1: The mapping approaches: (a) one-to-many, (b) synchronized many-to-many, and (c) unsynchronized many-to-many. Each rectangle represents a single mocap frame and each ellipse represents a single audio descriptor frame. Connected frames represent consecutive frames.

put musical patterns, and therefore gives few opportunities for generating novel movement patterns.

7.3 Proposed Approaches

In this paper, we aim to learn the relationships between low-level audio features and movement parameters for the continuous synthesis of full-body movements in an unsupervised manner. With GrooveNet, we are investigating several directions to address this problem. This involves the pipeline of the system, the choice of a suitable machine learning model, as well as different methods for representing the audio data.

Pipeline

Three strategies for mapping audio data to motion capture (mocap) data are illustrated in Figure 7.1: (a) one-to-many mapping, (b) synchronized many-to-many mapping, and (c) unsynchronized many-to-many mapping. While in all approaches the model takes a sequence of mocap frames as input, they differ in how the audio descriptors are involved. In a one-to-many mapping approach, the audio descriptor at time t together with the input mocap history determines the generated mocap frame at time t. In a synchronized many-to-many mapping, for each mocap frame, there exist an audio descriptor. The model takes a sequence of mocap history corresponding to the frames at the time interval of $[t-N, t-1]$, and a sequence of audio descriptors, with same length as the history, and generates the output mocap frame at time t. In an unsynchronized many-to-many mapping, the model takes the mocap history and the audio descriptor sequences that have different lengths, and it is up to the model to determine their temporal correlations. In this paper, we present a model following the one-to-many mapping approach.

Machine Learning Model

The two machine learning models that we are employing are Factored Conditional Restricted Boltzmann Machines (FCRBMs) and Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN). FCRBM has shown to be a suitable choice for movement generation, in particular to allow for a fine control over the generated movements (Taylor and Hinton, 2009; Alemi et al., 2015) and because it can generalize over the space of variations. The LSTM-RNN is powerful to model time series with complex temporal structures, and was shown efficient for controlled character and hand-writing generation (Graves, 2013), as well as uncontrolled dance movement generation (Crnkovic-Friis and Crnkovic-Friis, 2016).

Our initial experiments show that compared to the LSTM-RNN, it is easier to train the FCRBM on real-valued, continuous data. Also, FCRBM works better on smaller training sets and it is faster during generation. While FCRBM works better for the one-to-many mapping approach, the LSTM-RNN is more suitable for many-to-many approaches. As our initial experiments have not been successful with the LSTM-RNN yet, in this paper we only report the experiments using FCRBM.

Audio Representation

With respect to representing audio data, we follow two different approaches: 1) feature extraction and 2) feature learning. We describe our approach to audio feature extraction in Section 7.4. For feature learning, we have recently started training GrooveNet on audio features based on the temporal embeddings from a WaveNet-style auto-encoder (Engel et al., 2017), which is trained on raw audio from musical instrument sounds. In this paper, we only report the results from training GrooveNet with the extracted features.

7.4 Dataset and Feature Extraction

Few motion capture datasets include dance movement data. To our knowledge, no dataset of synchronized music and motion capture data is currently available online. We created a dataset containing four performances of a dancer. The music tracks are made by Philippe Pasquier and Philippe Bertrand at the Robonom sound studio in France using the StyleMachine lite from the Metacreative Technologies company[1]. We used three of the generated songs that belong to the genre of electronic dance music, with a regular tempo varying between 125 and 135 beats per minute.

The dancer's movements were captured using a 40-cameras Vicon optical motion capture system.The motion capture data was post-processed and synchronized with the audio data. The resulting dataset contains about 23 minutes of motion capture data recorded at 60

[1] https://metacreativetech.com

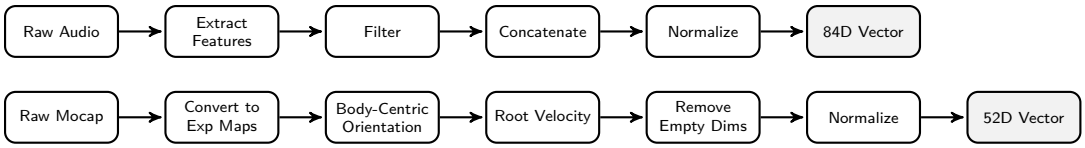


Figure 7.2: Overview of GrooveNet's data processing pipeline for the audio and movement modalities.

frames-per-second, giving a total of 82151 frames. We captured a total of four sequences, with two sequences dancing to the first song, and two sequences dancing to the second and the third song.

7.4.1 Audio Data Representation and Feature Extraction

Our goal is to generate movement in real-time from an audio stream. To that end, the audio signal must be represented by a sequence of features that describe the acoustic properties of the music continuously, with a similar temporal density as the movement data.

For each audio file, we extracted a set of low-level features at the same framerate as the motion capture data. We used a standard set of features described in the music information retrieval literature (Peeters, 2004; Bogdanov et al., 2013), including low-level features (RMS level, Bark bands), spectral features (energy in low/middle/high frequencies, spectral centroid, spectral spread, spectral skewness, spectral kurtosis, spectral rolloff, spectral crest, spectral flux, spectral complexity), timbral Features (Mel-Frequency Cepstral Coefficients, Tristimulus), melodic Features (pitch, pitch salience and confidence computed with the YIN algorithm (De Cheveigné and Kawahara, 2002), inharmonicity, dissonance). The features were computed using the Essentia open-source library (Bogdanov et al., 2013), with a window size of 66.7 ms and a hop size of 16.7 ms. We feature sequences were filtered with a FIR low-pass filter with a cutoff frequency of 5 Hz, in order to guarantee a smooth evolution of the audio descriptor that matches the time scale of dance movements. The resulting sequences were synchronized with the motion capture data and contain 84 dimensions (Figure 7.2-top).

7.4.2 Motion Capture Data Representation

The original motion capture data uses a skeleton with 30 joints, resulting in 93 dimensions including the root position, with their rotations represented in Euler angles. The data is recorded at 60 frames-per-second. We converted the Euler angle representations to exponential maps (Grassia, 1998) to avoid loss of degrees-of-freedom and any discontinuities. We removed the empty and fixed dimensions of the data. We also replaced the root's global orientation with the rotation velocity along the axis that is perpendicular to the floor plane, and replace the root's global translation with the 2-dimensional velocity of the root as pro-

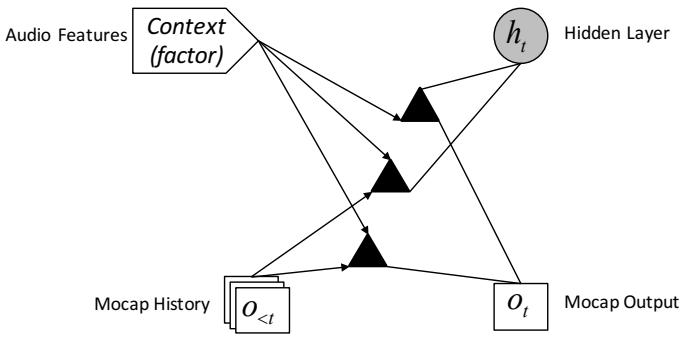


Figure 7.3: The architecture of a Factored CRBM with audio features fed into its context unit and mocap feature to its output/visible units.

jected on the floor plane. The resulting dataset contains 90151 frames, each represented by a 52-dimensional vector (Figure 7.2-bottom).

7.5 The Machine Learning Processes

7.5.1 Factored Conditional Restricted Boltzmann Machines

We use a Factored Conditional Restricted Boltzmann Machine (FCRBM) (Taylor and Hinton, 2009), shown in Figure 7.3, as the underlying machine learning model of GrooveNet. FCRBM is an energy-based generative model that learns to predict its output given a sequence of input data, modulated by its context data. Using a set of three multiplicative gates, the values of the context unit modulate the weights between the condition units (history data), the hidden units, and the output visible units. This arrangement allows the context data to directly, and in a non-linear way, control the network's output by manipulating the networks energy landscape.

FCRBM supports a multi-dimensional discrete or continuous context variable, which allows this model to capture and represent different qualities and semantics of human movement. Furthermore, one can interpolate or extrapolate the context values in order to create new movements that did not exist in the training data. In GrooveNet, we feed the audio features to this context unit in order to let the model to learn the relationship between the audio features and the dynamic processes behind the movements in the training data.

7.5.2 Learning

We train the model to predict the next motion capture frame at time t, given a recent history of the motion capture frames, $[t - N, t - 1]$, where N is the order of the model, representing the number of past motion capture frames to include in the prediction. The prediction is modulated by the a single frame of audio features at time t, fed to the context unit.

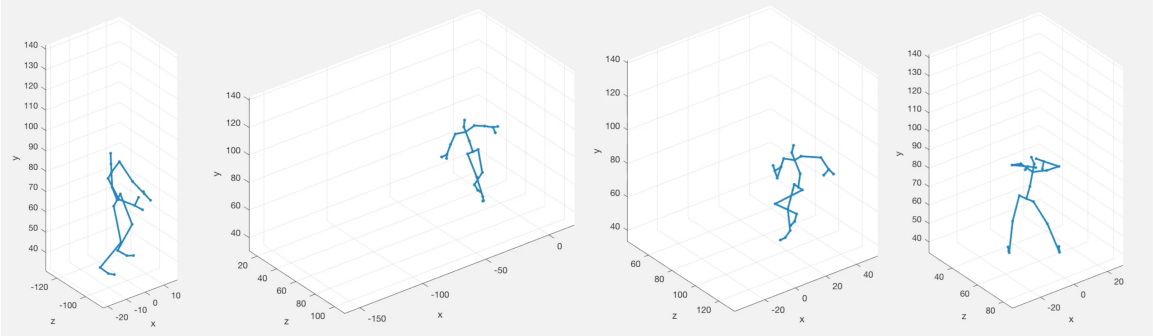


Figure 7.4: Some still frames from the generated movement patterns.

7.5.3 Generation

The generation is done using an iterative sampling process. The model predicts one frame of the movement at a time, given previous frames of movement and the audio features. It then uses the newly generated frame of movement as part of the input for predicting the next frame and continues the generation process.

7.6 Preliminary Results and Discussion

In this section, we present the preliminary results of GrooveNet. All the video outputs of the results are available on the accompanying web-page of the paper at: `http://omid.al/groovenet-material-ml4c/`.

7.6.1 Learning and Generating Movement Patterns

We start our investigation by learning the individual dance patterns that exist in our training data to see if the model can generate the patterns independently of the audio data. To this end, we manually manually segmented the dance sequences based on the main parts of the song, as illustrated in Figure 7.5. This annotation allows us to assess where the model can effectively encode consistent dance patterns. We then used a one-shot encoding scheme to label each pattern. Once the model is trained, we can then generate each pattern by feeding the desired label to the context unit of the FCRBM.

The results show that the FCRBM is able to learn the patterns from a very small training set (only one mocap sequence of about 4 minutes). As long as the same label is given to the model, it generates the same pattern continuously and repetitively, and changing the label will cause the model to transition to another pattern. still frames from the generated patterns for the first track are shown in Figure 7.4. The videos of the generated patterns are also available in the supplementary material.

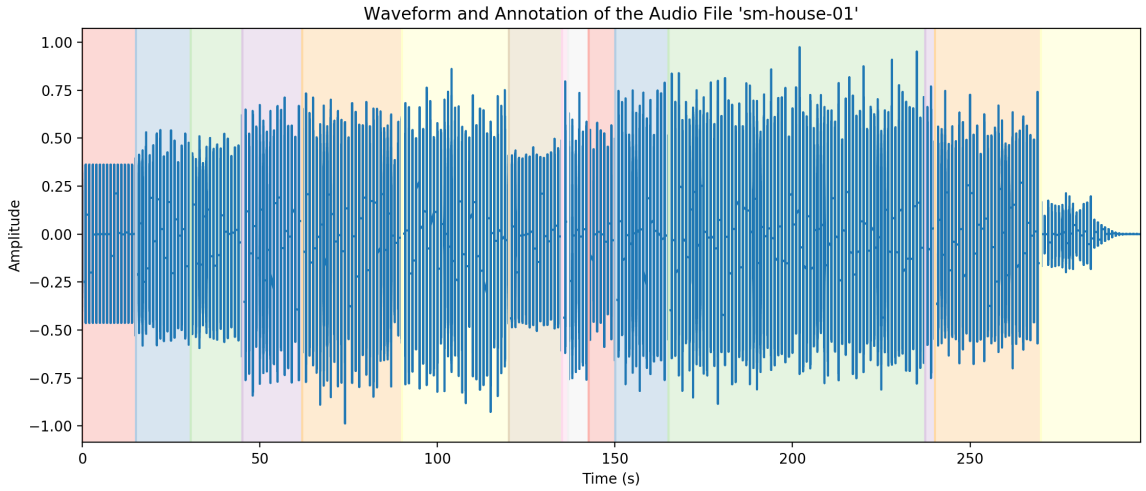


Figure 7.5: An example of manual segmentation of the first song used for preliminary experiments in generating movement patterns.

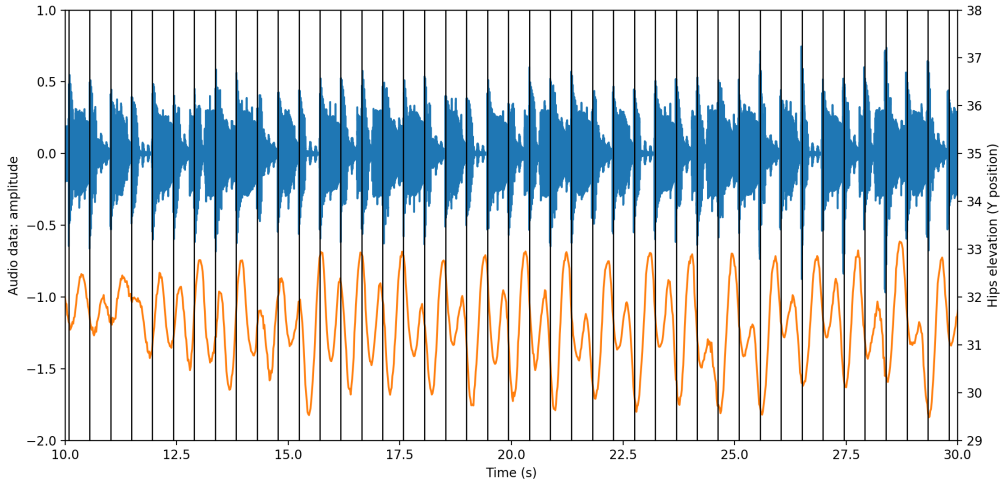


Figure 7.6: Visualization of the hips' position along the vertical axis (bottom) and the audio amplitude (top).

7.6.2 Dancing with Training Songs

In the previous experiment, we assessed the ability of the model to encode independent dance patterns from the manual annotation of the dataset. We now consider a fully unsupervised approach, where the model is trained with the entire dataset composed of four performances without any additional annotation. Our goal is to evaluate whether the model can learn the mapping between the audio features and the movement parameters on longer sequences, with a larger corpus of music and dance. As the first step in this direction, we assess the

movements generated by the model using as input data the songs already 'heard' during the training.

The results, as presented in the supplementary materials, show that the FCRBM is able to generate dance patterns consistent with the training set. Furthermore, the model captured the synchronization patterns between the rhythmic structure of the song and the generated movement (Figure 7.6). While the generated dance movements are plausible, we can note that the movements are at times jerky and can present artifacts such as foot sliding. The novelty in the generated movements remains to be further investigated.

7.6.3 Dancing with Unheard Songs

We evaluate the generalizability of the model, testing its performance on the songs that were not included on the training data. The results show that the FCRBM is not generalizing beyond the songs that exist in the training data. The three songs that are provided in the training

7.6.4 Computational Performance

A model with 500 hidden units, 500 factors, and an order of 30 past frames consists of 1,452,720 number of trainable parameters and it takes on average 0.0115 seconds to generate each frame on an Intel(R) Core(TM) i7-4850HQ CPU at 2.30GHz. This is fast enough to generate the movements at 60 frames-per-second in real-time.

7.7 Conclusion and Road Map

We presented the initial results from our GrooveNet project, in which we address the generation of dance movements in real-time from musical audio. This problem involves learning a cross-modal mapping between acoustic features and movement data, and generating dance movements from a new audio sequence.

We are investigating multiple audio-to-movement mapping approaches, machine learning models (FCRBM and GRU-RNN), and description methods of the musical information (features extraction vs feature learning).

Our preliminary analysis show that our model is able to learn and generate basic dance movements, independent of the audio data. In addition, it and learn and generate movements based on the song that it is trained with. However, the model currently fall short in generalizing beyond those songs in the training data and highly overfits. We believe that this is mainly due to our small and sparse training data set.

Bibliography

O. Alemi, W. Li, and P. Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*. 442–448.

B. Bläsing, B. Calvo-Merino, E. S. Cross, C. Jola, J. Honisch, and C. J. Stevens. 2012. Neurocognitive control in dance perception and performance. *Acta Psychologica* 139, 2 (2012), 300–308.

D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. 2013. Essentia: An Audio Analysis Library for Music Information Retrieval.. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR'13)*. 493–498.

M. Brand and A. Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 183–192.

B. Caramiaux, P. Susini, T. Bianco, F. Bevilacqua, O. Houix, N. Schnell, and N. Misdariis. 2011. Gestural Embodiment of Environmental Sounds : an Experimental Study. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

C. Chiu and S. Marsella. 2011. How to Train Your Avatar: A Data Driven Approach to Gesture Generation. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents*. Springer, 127–140.

L. Crnkovic-Friis and L. Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. *CoRR* abs/1605.06921 (May 2016).

A. De Cheveigné and H. Kawahara. 2002. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America* 111, 4 (2002), 1917–1930.

Y. Ding, M. Radenen, T. Artieres, and C. Pelachaud. 2013. Speech-driven eyebrow motion synthesis with contextual Markovian models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3756–3760.

C. Donahue, Z. C. Lipton, and J. McAuley. 2017. Dance Dance Convolution. (March 2017). arXiv:1703.06891

J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. (April 2017). arXiv:1704.01279

F. S. Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48.

A. Graves. 2013. Generating Sequences With Recurrent Neural Networks. (Aug. 2013). arXiv:1308.0850

D. Herzog, V. Krueger, and D. Grest. 2008. Parametric Hidden Markov Models for Recognition and Synthesis of Movements. In *Proceedings of the British Machine Vision Conference*. 163–172.

G. Hofer. 2009. *Speech-driven animation using multi-modal hidden Markov models*. PhD Dissertation. University of Edimburgh.

D. Holden, J. Saito, and T. Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics* 35, 4, Article 138 (July 2016), 11 pages.

S. Levine, P. Krähenbühl, S. Thrun, and V. Koltun. 2010. Gesture controllers. *ACM Transactions on Graphics* 29, 4, Article 124 (jul 2010), 11 pages.

Y. Li, T. Wang, and H.-Y. Shum. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM, 465–472.

K. Nymoen, B. Caramiaux, M. Kozak, and J. Tørresen. 2011. Analyzing Sound Tracings - A Multimodal Approach to Music Information Retrieval. In *Proceedings of the 1st International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies (MIRUM '11)*. 39–44.

F. Ofli, Y. Demir, Y. Yemez, E. Erzin, A. M. Tekalp, K. Balcı, İ. Kızoğlu, L. Akarun, C. Canton-Ferrer, J. Tilmanne, et al. 2008. An audio-driven dancing avatar. *Journal on Multimodal User Interfaces* 2, 2 (sep 2008), 93–103.

F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp. 2012. Learn2Dance: Learning Statistical Music-to-Dance Mappings for Choreography Synthesis. *IEEE Transactions on Multimedia* 14, 3 (June 2012), 747–759.

G. Peeters. 2004. *A large set of audio features for sound description (similarity and classification) in the CUIDADO project*. Technical Report. Icram.

A.-A. Samadani, E. Kubica, R. Gorbet, and D. Kulić. 2013. Perception and Generation of Affective Hand Movements. *International Journal of Social Robotics* 5, 1 (2013), 35–51.

G. W. Taylor and G. E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, 1025–1032.

J. Tilmanne and T. Dutoit. 2010. Expressive gait synthesis using PCA and Gaussian modeling. In *Proceedings of the Third international conference on Motion in games*. Springer-Verlag, 363–374.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 975–982.

Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. 2005. Learning hierarchical non-parametric hidden Markov model of human motion. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*. IEEE, 3315–3320.

E. Yamamoto, S. Nakamura, and K. Shikano. 1998. Speech-to-lip movement synthesis based on the EM algorithm using audio-visual HMMs. In *Proceedings of the Second Workshop on Multimedia Signal Processing*. 2–5.

T. Yamazaki, N. Niwase, J. Yamagishi, and T. Kobayashi. 2005. Human Walking Motion Synthesis Based on Multiple Regression Hidden Semi-Markov Model. In *Proceedings of the International Conference on Cyberworlds*. IEEE, 445–452.

Chapter 8

GrooveNet 2.0: Music-Driven Dance Generation

Abstract

We present *GrooveNet 2.0*, a system that learns to generate grooving dance movements by "listening" to an audio stream in real-time. Developing movement generation algorithms that are controlled by complex stimuli in real-time is a challenging problem. In particular, dancing in response to music represents an intricate human behaviour to model as it involves learning complex and non-linear mappings between two temporally and spatially different patterns. To address this problem, we use a recurrent neural network with gated recurrent units (RNN-GRU) to jointly learn the dynamics of movement and the cross-modal relationship between the audio and movement data. The key idea is to use both the movement and audio features as the inputs to the network and train the model to predict the next movement frame. In this way, we push the model to learn how the audio features influence movement dynamics. We train the network on a small dataset of synchronized motion capture and audio data of performers dancing on electronic dance music. We demonstrate the capabilities and limitations of our system in generating dance movements based on music from a variety of music genres.

Supplementary Material: All the videos, training data, and the source code are available at: ***https://omid.al/phdthesis#groovenet2***

8.1 Introduction

Real-time generation of human movement in the form of skeletal animation is a challenging and open problem. This problem becomes more challenging when attempting to automatically direct and control the generated movements' characteristics for a given stimulus with modality and spatial and temporal qualities different from the movement itself. For example, the movement can be controlled based on low-dimensional and temproally coarse labels describing the mover's emotional state (Alemi et al., 2015) or control cues describing the points of contact between the mover and its environment (Starke et al., 2019), or based on a high-dimensional, temporally dense stream of audio as in speech (Chiu and Marsella, 2011b) or music (Alemi et al., 2017).

In this work, we focus on dancing and informal choreography in response to music. Dancing is a creative and expressive form of human movement. It involves both cognitive and sensorimotor faculties, including elaborate motor control, precise timing and synchronization, and memory to name a few. When dancing to music, the outcome depends on the dancer's skills and interpretation of the music, as well as her or his personal movement signature and therefore the outcome differs from dancer to dancer (Alemi and Pasquier, 2019). Together, these make dancing a complex human behaviour to computationally model, and music-driven dance generation a complex case of cross-modal mapping between two sequences.

Music-driven dance generation involves complex and non-linear mappings between two temporally and spatially different patterns. A solution to build models of such mappings is to use data-driven and machine learning approaches to learn the mapping from recorded examples. To this end, we present *GrooveNet 2.0*, a multi-modal approach for real-time generation of grooving-like dance movements based on a given music stream. Our main objective for GrooveNet 2.0 is for it to generate dance moves in a human-like manner while following the music's rhythm and temporality. We train *GrooveNet 2.0* on a dataset of synchronized music represented in raw audio (as opposed to symbolic music) and dance recordings in the form of motion capture data representing 3D skeletal animation (as opposed to 2D or video-based pose representations). Once trained, *GrooveNet 2.0* can "listen" to music and generate movements in real-time without the need to use or retain the training data.

GrooveNet 2.0 uses a recurrent neural network (RNN) at its core in a relatively compact architecture. As a result, GrooveNet can efficiently generate movements in real-time on a CPU-only system. Unlike most of the RNN-based movement generation works that suffer from the "dying out" effect when generating long sequences (Jain et al., 2015; Fragkiadaki et al., 2015; Martinez et al., 2017; Alemi and Pasquier, 2019), GrooveNet can generate continuous streams of movement.

In the rest of the paper, we first provide a review of the related works. We then present our training data and detail the encoding schemes for both movement and audio features.

Next, we present our machine learning model and the training and generation processes. In the results section, we present and discuss various experiments on the generative capabilities of *GrooveNet 2.0* and demonstrate the output of the system based on a variety of music tracks from different genres. We examine how well the generated movements are synchronized with the input audio track, as well as the stylistic and aesthetic qualities of the movements.

8.2 Related Work

In this section, we first review the approaches in using machine learning models and 3D motion capture data to generate movement animation. Next, we review the works on dance generation. Finally, we review works that involve using audio features such as speech and music to control movement animation generation.

8.2.1 Machine-Learning-Based Movement Generation

Machine learning techniques such as dimensionality reduction, Gaussian Processes, Hidden Markov Models (HMM), and artificial neural networks are used for learning and generating 3D skeletal movement animation.

Dimensionality reduction techniques are used to generate movement by reversing the direction of the mapping between the high-dimensional and the manifold spaces (Tilmanne and Dutoit, 2010; Samadani et al., 2013; Herrmann et al., 2017). However, these approaches do not directly model the dynamics of movement and need extensive pre-processing steps that involve creating a fixed-length and aligned representation of the data. Gaussian Processes are also used to learn and generate movement (Wang et al., 2007; Levine et al., 2012; Taubert et al., 2012), but are slow during both the training and generation processes. Hidden Markov Models (HMMs) learn the dynamics of movement and can be used for real-time generation applications, but are prone to learning an average model due to their limited expressive power (Brand and Hertzmann, 2000; Herzog et al., 2008; Tilmanne et al., 2012).

A variety of neural network techniques and architectures have been used for movement generation. Neural networks can directly model movement dynamics, provide a better expressive power than HMMs, and, depending on their size and architecture, can be used for real-time generation. Factored Conditional Restricted Boltzmann Machine (FCRBM) is an energy-based neural network that is used to control and generate walking movements in real-time (Taylor and Hinton, 2009; Alemi et al., 2015). Hierarchical FCRBM extends FCRBM by adding another layer under the FCRBM layer to improve its interpolation and generalization capabilities (Chiu and Marsella, 2011a,c).

Holden et al. (2016) use convolutional autoencoders to learn a latent representation of movement using one-dimensional convolutions over the time dimension. A feed-forward convolutional network is then trained based on the latent representation of the data to learn

a mapping from control parameters to the output poses. Li et al. (2018) use a multi-layer convolutional architecture to capture the temporal characteristics of movement hierarchically. As the first layer captures the short-term dependencies between adjacent frames, higher-level layers capture dependencies between the lower-level ones, and thus enabling the architecture to learn long-term interactions.

Crnkovic-Friis and Crnkovic-Friis (2016) use a three-layer RNN with Long Short-Term Memory units (LSTM-RNNs) to learn and generate dance movements in an unsupervised and uncontrolled manner. Fragkiadaki et al. (2015) introduce a recurrent architecture that uses non-linear encoder and decoder networks connected to the inputs and outputs of LSTM units for learning and generating everyday movements. Martinez et al. (2017) use a recurrent sequence-to-sequence architecture with Gated Recurrent Units (GRU) that is common in machine translation models (Sutskever et al., 2014) to model every-day movements. Pavllo et al. (2018) present QuaterNet, a two-layer RNN architecture with GRUs. While the input and output of QuaterNet is represented by the joint rotations encoded in quaternions, during the training the loss function uses the joint positions. Jain et al. (2015) use multiple RNNs in a modular graph to model spatio-temporal data. The nodes of the graph represent the modules of the problem and the edges represent the spatio-temporal relationships between the modules.

Wang et al. (2019) combine RNNs with adversarial training techniques used by Generative Adversarial Networks (GANs). The proposed architecture consists of a recurrent movement model and a GAN model. Movements are generated by the movement model and are then refined through the generator network of the GAN network. Another group of works takes a different approach in using neural networks to generate movement by using Phase-Functioned Neural Network (PFNN) (Holden et al., 2017; Starke et al., 2019). The core idea of PFNN is to use a simple fully-connected neural network to generate a new frame of mocap data, while the weights of the network are themselves generated by a function that is based on a phase variable that represents the timing of a cyclic movement such as locomotion.

GrooveNet 2.0 uses an RNN-based architecture. The main difference between GrooveNet and the aforementioned RNN-based works is that GrooveNet is trained on multi-modal audio and mocap data and allows controlling the generation of the next mocap frame based on the given audio features.

8.2.2 Dance Movement Generation

Brand and Hertzmann (2000) and Ofli et al. (2012) use HMMs to generate dance movements, while Wang et al. (2005) train Hierarchical Hidden Markov Models with non-parametric output distributions (NPHHMM) to generate ballet and disco movements. Crnkovic-Friis and Crnkovic-Friis (2016) use Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) to learn and generate contemporary dance movements in an unsupervised and un-

controlled manner. Donahue et al. (2017) train LSTM-RNNs on step charts from the Dance Dance Revolution game choreographies. Pettee et al. (2019) train autoencoders based on an encoder-decoder LSTM architecture on dance data. The authors use this model to create novel choreographies or create variations of an input choreography by drawing or manipulating curves that are defined in the latent space of the autoencoder.

Our approach differs from the aforementioned dance generation work in that GrooveNet controls the dance generation using music features and in real-time.

8.2.3 Audio-Driven Movement Generation

We divide the literature on controlling the movement generation based on audio signals into speech-driven and music-driven. While both speech-driven and music-driven movement generation take raw audio signals as input, there are fundamental differences between the two. When speaking, both the acoustic and motion signals are created by the same underlying processes within an individual and work together to convey the same information. When dancing to music, the relationship between the two modalities presents a complex hierarchy of temporal structures. For example, grooving music is structured by the variations and repetition of rhythmic, melodic, and harmonic patterns. When dancing to such music, the movement also constitutes patterns that tend to follow the musical patterns with variations. Furthermore, the outcome depends on the genre of the music, the context, and the expertise and personal characteristics of the performer. We first briefly review speech-driven generation works

Speech-Driven Movement Generation. HMMs are used to generate the movements of individual body parts such as the head (Hofer, 2009) or eyebrows (Ding et al., 2013) based on speech. Levine et al. (2010) use Hidden Conditional Random Fields for generating speech-driven hand gestures. Chiu and Marsella (2011c) use Hierarchical FCRBM to control gestures matching the prosody of speech.

Music-Driven Dance Generation. Ofli et al. (2008) use HMM to generate beat-synchronized dance movements. In their approach, the mocap data is first segmented based on the measures in the accompanying music. The model is then tasked to learn a mapping between specific music patterns in each measure and a movement segment, which limits the ability of the model to generate novel dance movements. Lee et al. (2018) adapt the Dilated Convolution model from text-to-speech generation literature (Tachibana et al., 2017) to generate music-driven dance movements. Their encoder-decoder architecture consists of two encoder networks, one for audio and one for movement data, and a single decoder network that only outputs movement data. They train the model on a dataset of 2-dimensional poses extracted from Youtube videos of K-pop music videos. Alemi et al. (2017) create a real-time dance generation system by training an FCRBM on a dataset of motion capture data. With FCRBM, the music features explicitly modulate the weights of the network, in effect influencing the behaviour of the network and controlling the generation (Taylor and

Music Track	No. of Movers	BPM	Music Length (mm:ss)	Captured Length (mm:ss)	Credits
LikeMyJack	10	128	4:58	42:10	*
Roucoule	10	128	7:02	32:27	*
sKi	10	135	8:10	33:36	*
Inna	9	126	6:07	31:36	*
Chasing the Learning Rate	6	125	3:07	18:44	§
Libra Mix 100 excerpt	9	103	3:00	16:09	†
Libra Mix 97 excerpt	1	132	1:30	1:30	††
Bend	8	118	7:09	30:45	‡
				03:21:02	

Table 8.1: List of Dance-Music Data. The data is available at `https://movehub.omid.al/groovedb/`. The *Music Length* column shows the length of the music track while the *Captured Length* column shows the total length of the captured data based on this music track from all of the performers. Note that the duration of recordings for each track differs from performer to performer. Music credits: *: Philippe Pasquier and Philippe Bertrand at the Robonom sound studio in France using the StyleMachine lite by Metacreative Technologies Inc. §: Philippe Pasquier and Renaud Bougueng Tchemeube at the Robonom sound studio in Vancouver using the MMM model (Ens and Pasquier, 2020) in Apollo (). †: By Doc Jay - From `https://libramix.org/`. ††: By Andry - From `https://libramix.org/`. ‡: By Grant.

Hinton, 2009). While their approach can generate movements based on the songs that exist in the training data, it fails to generalize to novel songs.

GrooveNet 2.0 differs from the aforementioned music-driven dance generation approaches in that it does not rely on classification or segmentation of the audio signal, is trained on high-quality 3-dimensional motion capture data, and is able to generalize beyond the music that exist in the training data.

8.3 Data

In this section, we start by describing the contents of our training dataset. Next, we describe the procedures we perform on the raw training data to prepare them for training the model. This includes changing the representation of the motion capture data, extracting audio features, and creating fixed-length sequences for mini-batch training.

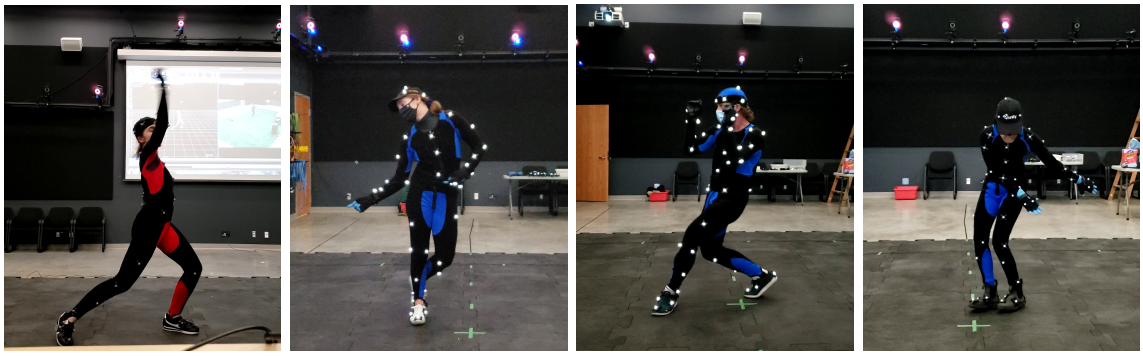


Figure 8.1: Photos from some of the performers in the GrooveDB

8.3.1 Dataset

We created a dataset containing synchronized motion capture and music pairs. For each pair, a performer dances to the music being played in a motion capture studio. We recruited ten performers (six female, four male) and instructed them to groove with the beats, but did not give them any specific choreography to follow. Some photos of the performers are shown in Figure 8.1. The music tracks in the training set are electronic dance music with a regular tempo, as outlined in Table 8.1. The motion capture data is represented by a skeleton with 30 joints.

8.3.2 Motion Capture Data Representation

Motion capture data are typically structured around a virtual skeleton defined as a hierarchy of joints. The skeleton hierarchy starts with a *root* joint that represents the global position and orientation of the skeleton. The configuration of the body is then defined by the rotation of the rest of the joints in the skeleton parameterized in Euler angles.

Using a hierarchy of joint rotations has become the standard practice in computer animation due to its advantages over using joint positions. First, using joint rotations ensures that body limbs have fixed lengths throughout the movement. Second, using a virtual skeleton makes it easier to use the data to rig animation characters as it is more convenient to position the animated character in different locations and orientations in the scene.

Different numerical requirements of machine learning algorithms have resulted in using alternative methods such as positions, quaternions, and exponential maps to represent the motion capture data (Alemi and Pasquier, 2019). While the rotation representation is suitable for animation, rotations are susceptible to singularities, non-unique representations, and discontinuities, which make it harder for machine learning models to learn the patterns in the data. An alternative approach is define the loss function over the joint positions, which avoid the aforementioned problems. We experiment with both rotation and position representations. In the following, we describe the steps we take to prepare the mocap data for training.

Joint Rotation Representation. For representing joint rotations, we convert the Euler angles to exponential maps (Grassia, 1998) and convert the global position and orientation to a location-and-orientation-invariant representation described by Taylor (2009).

Joint Position Representation. Each mocap frame in the training data describes the agent's pose through the absolute positions of the body joints in a global coordinate system. In this representation, the position of each joint is independent from the position of other joints. We denote each mocap frame $\mathbf{m}$ of this representation at time t as follows:

$$\mathbf{m}^{(t)} = [p_x^{root}, p_z^{root}, p_y^{root}, p_x^{spine}, p_z^{spine}, p_y^{spine}, ...]$$

We first transform the mocap representation to be invariant to the absolute location and orientation of the body so that the same pose in different locations and orientations will be represented with the same values. Note that to do so we only need to manipulate the joint positions on the ground plane (xz) as their vertical position (y) in invariant to the global location and orientation of the body (e.g., for the same pose, the distance of the neck joint from the ground plane is the same no matter where the character is located on the ground plane and which direction it is facing).

To this end, we define a 2-dimensional, body-centric local coordinate system that moves and re-orients itself at each frame. The origin of this local coordinate system is at the position of the root joint, the lateral axis is defined along the left and right hip joints, and the sagittal axis is defined along the vector passing through the origin and is perpendicular to the lateral axis.

For each frame, we transform the absolute position of every joint (except the root joint) to this body-centric local coordinate system by creating a transformation matrix that rotates and translates the joint positions along the ground plane. Considering θ being the angle between the x axis of the global coordinate and the lateral axis of the body, and using the hat ($\hat{\bullet}$) notation to denote the transformed counterparts of variables, we perform the following transformation on every joint j:

$$\begin{bmatrix} \hat{p}_x^j \\ \hat{p}_z^j \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & p_x^{root} \\ sin(\theta) & cos(\theta) & p_z^{root} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x^j \\ p_z^j \\ 1 \end{bmatrix}$$

Next, we replace the position of the root joint on the ground plane p_x^{root} and p_z^{root} by its velocity vectors Δ_x^{root} and Δ_z^{root}. Finally, we normalize all of the values to have zero mean and unit standard deviation. We denote the resulting 84-dimensional mocap feature vector $\hat{\mathbf{m}}$ for frame t as follows:

$$\hat{\mathbf{m}}^{(t)} = [\Delta p_x^{root}, \Delta p_z^{root}, p_y^{root}, \hat{p}_x^{spine}, \hat{p}_z^{spine}, p_y^{spine}, ...]$$

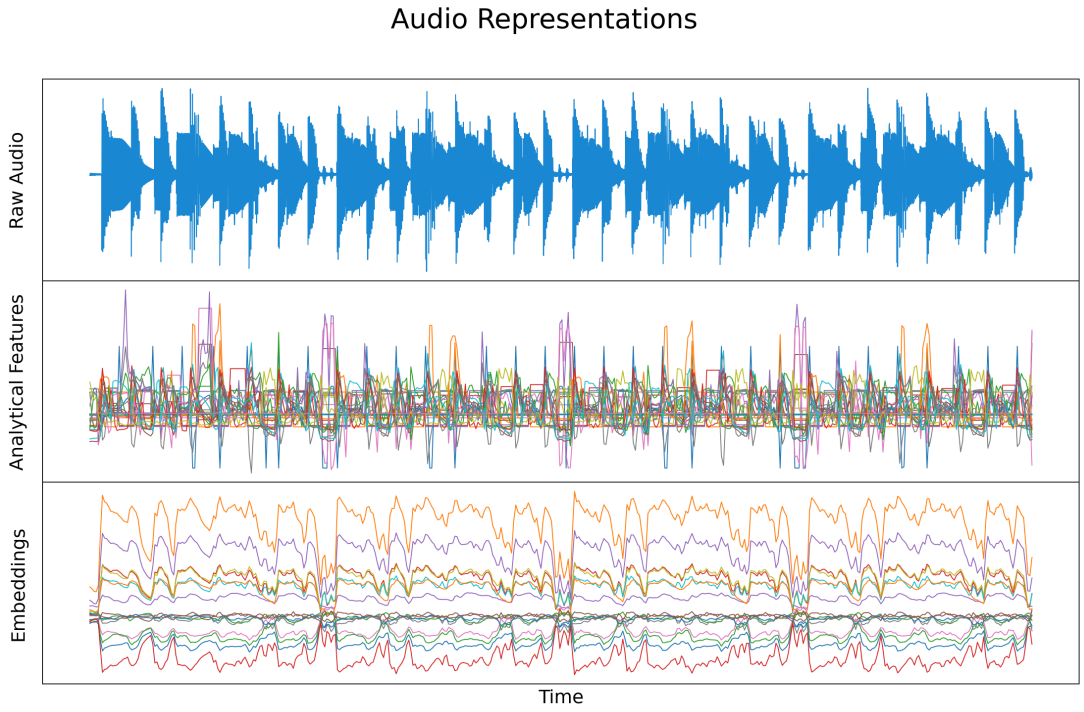


Figure 8.2: Different audio representations. Top: the raw audio signal. Middle: analytical features. Bottom: embeddings.

After generating a new sequence by the model, we simply reverse this procedure and reproduce the absolute joint positions.

8.3.3 Audio Representation

We process the raw audio signal to create a sequence of features that describe the music. We experiment with two different approaches to represent music: 1) using analytical features extracted from the audio signal and 2) using latent representations (embeddings) learned by another neural network. A plot of each representation is shown in Figure 8.2. We describe our approach to both in the following.

Analytical Features

We consider a set of features taken from the music information retrieval literature (Peeters, 2004a; Bogdanov et al., 2013). As shown in Table 8.2, the features describe the low-level, rhythmic, spectral, and timbral qualities of the audio signal. We extract the audio features from the raw audio signal using a sliding window method. The hop size is defined as the ratio of the audio sample rate (44100 Hz) and the mocap framerate (30 FPS). This hop size effectively down-samples the audio features to the same framerate as the mocap data. We then normalize the values to have a zero mean and unit standard deviation.

Group	Feature	Description
Rhythmic	Beat Signals	A square wave signal built from the estimated position of the beats (Degara et al., 2012)
	Beat Loudness	Spectral energy of each beat segment across the whole spectrum
Low-level	RMS Level	Root mean square of the audio signal
Spectral	Energy in low/middle/high frequencies	The energy[1] of the audio signal, calculated for the whole spectrum and for frequency bands of $[20Hz, 150Hz]$, $[150Hz, 800Hz]$, $[800Hz, 4kHz]$, and $[4kHz, 20kHz]$
	Spectral Centroid, Spectral Spread, Spectral Skewness, and Spectral Kurtosis	The 0th, 1st, 2nd, 3rd and 4th central moments[2] of the audio signal
	Spectral Roll-Off	The roll-off frequency which can be used to differentiate between the harmonic and noisy signals[3]
	Spectral Crest	A measure of the noisiness of the signal (Peeters, 2004b)
	Spectral Flux	The L2-norm of the changes in magnitude (Dixon, 2006)
	Spectral Complexity	The number of peaks in the audio signal
Timbral	Tristimulus	Three energy ratios describing the first harmonic of the spectrum (Peeters, 2004b)

Table 8.2: The analytical audio features extracted from music.

In our experiments, we use a window size of 66.7 ms, a hop size of 1470 audio samples. The final audio feature vector contains 30 dimensions. All the feature extraction operations are performed using analytical methods provided by the Essentia open-source library (Bogdanov et al., 2013).

Embeddings

We also experiment with training the model with an embedding representation of the audio signal using a pre-trained NSynth model that uses a WaveNet-style autoencoder to learn a low-dimensional, temporal representation of audio signals (Engel et al., 2017). Because of the autoencoder architecture, the 16-dimensional embeddings can be used to reconstruct the audio signal. We hypothesize that by using an embedding representation GrooveNet will be better able to generalize over unheard songs.

8.3.4 Creating Training Mini-Batches

So far, our training data set contains pairs of synchronized music and mocap feature vectors with different lengths. Since our machine learning model takes both modalities as input, we concatenate the music and mocap data to have a combined feature vector. We then segment the combined feature vectors of arbitrary lengths into fixed-length sub-sequences to create the mini-batches that we use for batch training.

8.4 Learning to Generate Dance Movements

We use a recurrent neural network to jointly learn the dynamics of audio, movement, and the cross-modal relationships between audio and movement data. A recurrent network works by sharing its parameters across different time steps. As a result, the same network is used to process the inputs at each time step, with an extra input from its own hidden state from the previous time step. This makes the hidden state of an RNN a function of its hidden state from the previous time step, which itself is a function of the hidden states from previous time steps. Thus, it is the hidden state that carries the information across all time steps and acts as a lossy memory-like cell, enabling the model to learn the long-term characteristics of an arbitrary-length input sequence. There are different memory cells introduced for RNNs, notably the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Graves, 2013) and the Gated Recurrent Units (GRU) (Cho et al., 2014). We choose to use GRUs for

[1] `https://en.wikipedia.org/wiki/Energy_(signal_processing)`

[2] `https://en.wikipedia.org/wiki/Central_moment`

[3] `https://essentia.upf.edu/reference/streaming_RollOff.html`

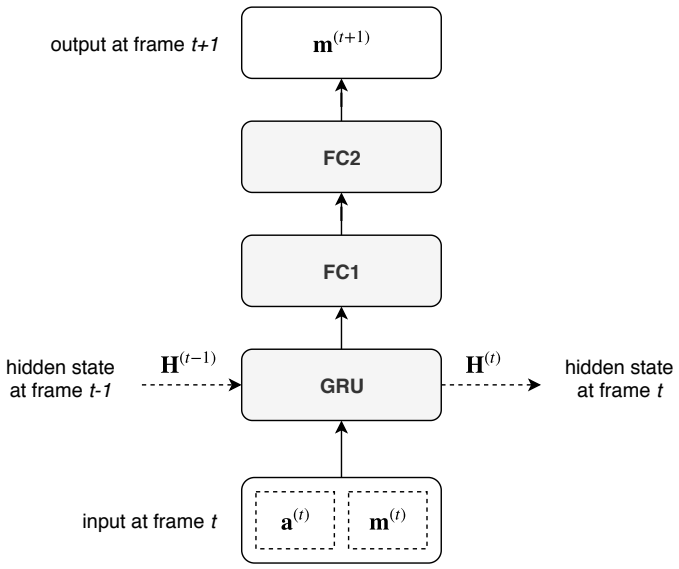


Figure 8.3: The machine learning architecture of *GrooveNet 2.0*. The GRU-RNN learns to predict the next mocap frame given the audio and mocap frames at the current time step as well as its hidden state from the previous time step. The output of the RNN is passed through a fully-connected layer with sigmoid (σ) activations. Another fully-connected layer with no activations is used to map the output to the mocap features.

GrooveNet 2.0 as our experiments showed it to be more stable than LSTM in generating movement.

The machine learning model of *GrooveNet 2.0* is tasked with predicting the next mocap frame given the current combined audio and mocap frame as well as the hidden state of the RNN from the previous time step:

$$p(\hat{\mathbf{m}}^{(t+1)}|\hat{\mathbf{a}}^{(t)}, \hat{\mathbf{m}}^{(t)}, \boldsymbol{H}^{(t-1)})$$

By formulating the problem as shown above, we note that predicting the next mocap frame depends on the audio features as well as the mocap features and the hidden state of the RNN. This, in turn, allows us to control the movement generation by the given audio features.

The neural network architecture of *GrooveNet 2.0* is shown in Figure 8.3. After experimenting with different activation functions, RNN cells, and number of layers, we found the following configuration to work best with our data. The input to the network is our combined feature vector $(\hat{\mathbf{a}}, \hat{\mathbf{m}})$, which is fed to a single layer of an RNN with GRU cells and exponential linear activations (*elu*). At each time step, the RNN takes its state at the previous time step and the input features at the current time step and predicts the mocap features for the next time frame as its output. We pass the output of the RNN through a fully-connected network with sigmoid activation functions before using another fully-connected layer with no activations to produce the final mocap features. Our initial experiments with the model

showed that adding the two fully-connected network help stabilize the output of the RNN. We use 300 units for the RNN and 200 units for the first fully-connected output layer.

8.4.1 Training

We describe the details of our approach to train the model in the following.

Training Parameters: We train the network using Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0004. We did not use any gradient clipping.

Loss Function: We use the Mean Squared Error (MSE) of the model's predicted mocap features and the ground-truth data.

Dropout Regularization: During the training, we randomly set some of the values in the input vector to zero with a rate of 40%. The dropout is applied to both the audio and mocap features in the same way. Applying dropout to the input of the network helps the model to be more robust against previously generated frames that might be noisy and imperfect as the model learns to recover from minor failures.

Stopping Criteria: To determine when to stop the training process, we train the model until the loss value stops to change significantly for a number of consecutive epochs, while saving a snapshot of the model weights separately at each epoch. We then generate samples from multiple epochs and decide on which epoch produces the best results. Note that we cannot merely rely on the loss value as a measure of the quality of a model at each epoch. A smaller loss value is not often equal to a model with better generative qualities. The results presented in the next section are generated from a model trained for 250 epochs.

8.4.2 Generation

The model generates mocap features one frame at a time in an autoregressive manner. The generation process starts with an initial mocap frame, which can be a vector of zero or a vector of randomly generated values, and a vector of zeros as its hidden state. At each generation step, the model takes the input audio frame, the previous mocap frame, and the previous hidden state and generates the next frame and the updated hidden state, which will be used for the next generation step.

8.5 Results and Discussion

8.5.1 Generation Result

We evaluate the performance of *GrooveNet 2.0* and its behaviour under different conditions through a series of experiments presented in this section. As the generative quality of the model can be better assessed through video rendering of the movements accompanied with music, all of the outputs presented here are also available in video format on the paper's complementary website: `https://omid.al/phdthesis#groovenet2`.

The Effect of the Number of the Performers in the Training Data

We first evaluate how adding more performers to the training set changes the generative performance of the model. We discover that adding more performers pushes the model to predict an average pose. We hypothesize that this issue can be resolved by predicting the parameters of a mixture distribution as discussed in Section 8.6. The results presented in the following are based on a model that is trained on only one performer.

Analytical Audio Features Versus WaveNet Embeddings

We train two sets of models: one based on the analytical audio features extracted from the raw audio signal and one based on the embedding representation of the audio signal encoded by a pre-trained WaveNet-style autoencoder. For each of the following experiments, we provide the results with both models. Our qualitative observation is that the analytical features perform slightly better than the embeddings.

Generation from the Songs in the Training Set

As a way to establish a sanity test for the generative capabilities of *GrooveNet 2.0*, we task the model to generate movements based on the music it has "heard" during the training. The model is able to generate dance-like movements when it is given one of the training songs as its input. Second, the model does not simply repeat the training data. Even when the model is primed with the corresponding training data, the generated movement follows a different pattern than the movement it had seen in the training set. Videos of the generated results are available in `https://omid.al/phdthesis#groovenet2`.

Generation from the Songs Not in the Training Set

As the next experiment, we generate movements from music tracks that were not included in the training set, but nevertheless follow a similar style (i.e., dance music) as to the music in the training set. As shown in videos in `https://omid.al/phdthesis#groovenet2`, the model is able to generate movements for the music that it has not "heard" during the training.

Generation from the Songs with Different Styles

To further investigate the generative capabilities of *GrooveNet 2.0*, we generate movements from music tracks that not only were not included in the training set, but also have largely different styles as to the songs in the training set. Videos of the generated results are available in `https://omid.al/phdthesis#groovenet2`.

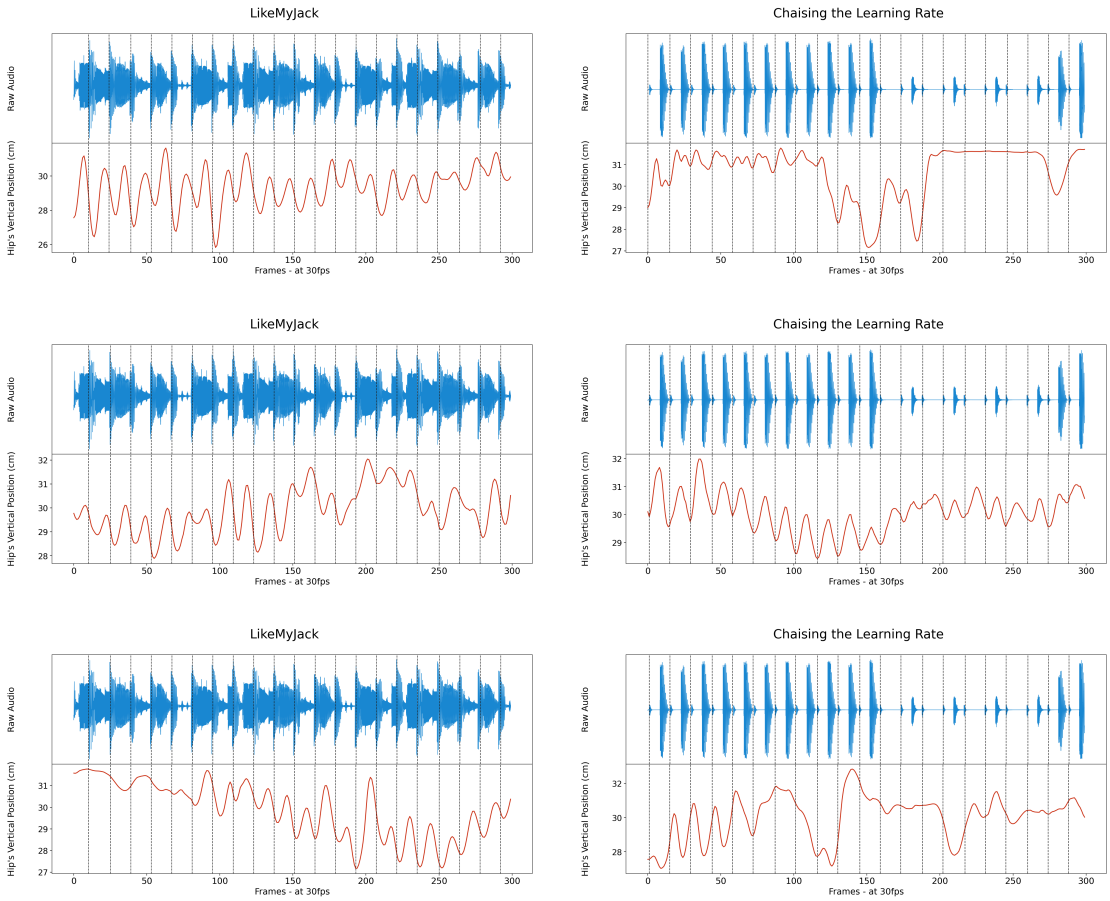


Figure 8.4: A visualization of the coordination between the rhythm of the input music and the rhythm of the generated movements using the songs in the training data. Top row: ground truth data. Middle row: with analytical audio features. Bottom row: with WaveNet embeddings.

Movement and Music Coordination

In this experiment, we look at whether the generated movements follow the rhythm of the music. We use the vertical position of the character's hips as an indicator of the rhythmic movements. We then plot the hip position, the input audio signal, and the estimated beat positions (as described in Section 8.3.3). We present the results in Figure 8.4 for the songs that are in the training set and in Figure 8.5 for the songs that are not in the training set.

Generation Speed

The model can generate movements at 160fps on an Intel(R) Core(TM) i7-8565U CPU 1.80GHz and 980fps on an NVIDIA GeForce RTX 2080 GPU.

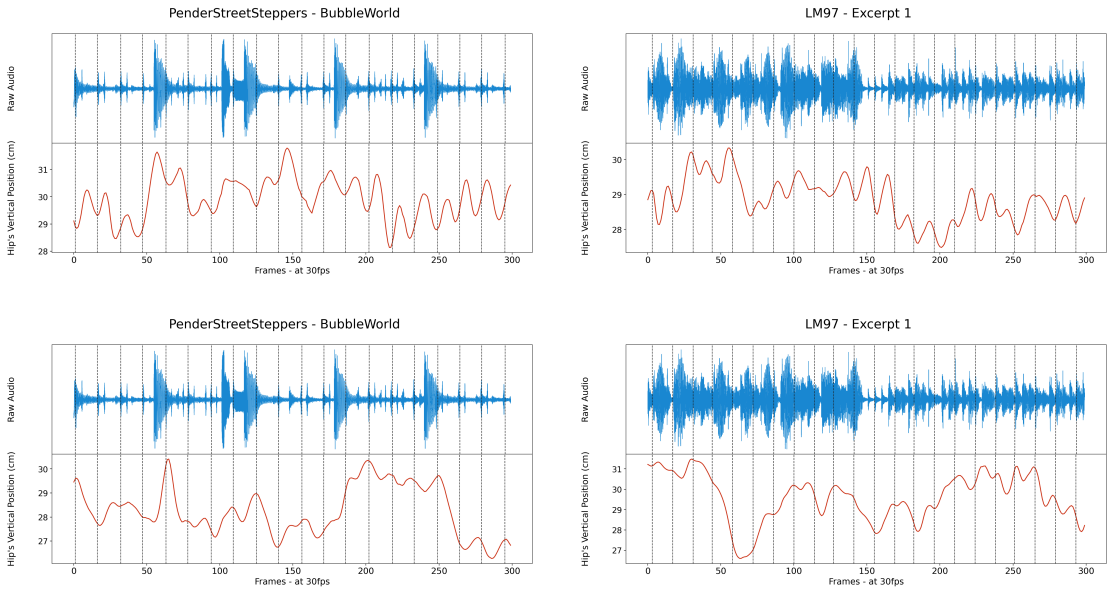


Figure 8.5: A visualization of the coordination between the rhythm of the input music and the rhythm of the generated movements using the songs not in the training data. Top row: with analytical audio features. Bottom row: with WaveNet embeddings.

8.6 Limitations and Future Work

The approach presented here has a number of limitations. In the following, we address each limitation and outline our proposed solutions for the future work.

Mixture Output. The current model performs poorly when trained on the data from more performers. We suspect that using a mixture model (e.g., Mixture Density Network (MDN) (Bishop, 1994)) at the output can increase the performance of the model when trained on the data from all ten human subjects in our dataset.

Model Output and Loss Function. Currently, we only use the ground-truth data during the training and use the Mean Squared Error (MSE) of the model's predicted mocap features and the ground-truth data to train the model. However, there are three issues with this setting.

Firstly, it is shown that training RNNs to predict real-valued data using an MSE loss function can push the model to predict an average value after a few iterations (Alemi and Pasquier, 2019). We plan on investigating a number of solutions such as extensions to the teacher forcing method to address this problem. For example, Li et al. (2017) alternate between the ground truth data and the past generated outputs of the network itself. In a similar approach, Pavllo et al. (2018) use a curriculum schedule technique (Bengio et al., 2015), which gradually increases the frequency of using the model's prediction over the ground-truth data. Another technique suggested in the literature (Martinez et al., 2017) is the DAGGER (Dataset Aggregation) algorithm (Ross et al., 2011). DAGGER iteratively

builds a dataset from the predictions at each iteration and uses the aggregated dataset for training in the next iteration.

Secondly, we are interested in implementing a differentiable forward-kinematic function that allows us to train the model on joint rotations, but to calculate the loss over the joint position space, as proposed by Pavllo et al. (2018).

Thirdly, with GrooveNet, we are not attempting to train the model in such a way that it predicts the exact movements that it has seen in the training data. As stated by Alemi and Pasquier (2019), because of the highly variable nature of movement, any loss function that penalizes divergence from the training data limits the creative performance of the model. While the predicted movement might be perceptually plausible, it might diverge from the ground truth numerically. We are interested in exploring loss functions that give more weight to factors such as how plausible the movement is or how well the generated movement follows the rhythm in the music. To this end, we will be exploring the adaptation of loss functions such as the Normalized Power Spectrum Similarity (NPSS) introduced by Gopalakrishnan et al. (2019) which targets the long-term predictive quality of generative models. form **Foot Sliding.** The generated movements suffer from foot sliding. In the future, we are interested in implementing contact-aware approaches as used in (Holden et al., 2017; Wang et al., 2019).

Artwork with GrooveNet. As an ongoing project, we are working on using GrooveNet in an interactive art installation in which the model generates movements based on the music provided by the audience. The generated movements are then used to drive audio-visual elements. To support this work, we have implemented a real-time system that generates movements based on any music that is being played on a computer. A video of the real-time dance generation in response to multiple songs is available in `https://omid.al/phdthesis#groovenet2`.

Controlling the Dancer's Style. For our artwork, we are interested in manipulating the style of the dances that are being generated by GrooveNet. When trained on the data from all ten performers in GrooveDB, we are looking at two approaches to achieve style manipulation: 1) using a one-hot encoding of the performer as extra input to the model, and 2) integrating a layer in the neural network architecture that learns a latent representation of the dancers' styles.

8.7 Conclusion

We present *GrooveNet 2.0*, a machine learning model for continuously generating dance movements in real-time given a continuous stream of music. We train the model on a multi-modal dataset of dance and electronic music tracks. The model is able to generate dance-like movements for a variety of music styles, even for styles that largely differ from those in the training set. *GrooveNet 2.0* learns the dynamics of human movement in response to music.

As a result, it has applications in areas where the embodied behaviour of an agent in the presence of audio has an important role. For example, we can use GrooveNet in areas such as the arts, video games, virtual reality, and product design industries to drive an agent's movements. The dancing agent's manifestation can be virtual (e.g., animation and visualizations) or physical (e.g., robots) and span from human-like avatars and non-human-like objects and abstract shapes. At the same time, *GrooveNet 2.0* learns the dynamics of music related to a dancer's movements. In the future, we are interested in creating models that can perform the reverse task by generating audio features from movement, which can be used in various artistic applications.

Acknowledgements

We thank Mirjana Prpa for her help with organizing the motion capture sessions. We also thank Maria Lantin, Richard Overington, and Sean Arden from Emily Carr University for giving us access to the motion capture studio at the Emily Carr University.

Bibliography

O. Alemi, J. Françoise, and P. Pasquier. 2017. GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks. Poster accepted to the Workshop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining - `https://ml4creativity.mybluemix.net/`.

O. Alemi, W. Li, and P. Pasquier. 2015. Affect-Expressive Movement Generation with Factored Conditional Restricted Boltzmann Machines. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*. 442–448.

O. Alemi and P. Pasquier. 2019. Machine Learning for Data-Driven Movement Generation: a Review of the State of the Art. *CoRR* abs/1903.08356 (2019). arXiv:1903.08356 `http://arxiv.org/abs/1903.08356`

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, 1171–1179.

C. M. Bishop. 1994. *Mixture density networks.* Technical Report. Aston University.

D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. 2013. Essentia: An Audio Analysis Library for Music Information Retrieval.. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR'13)*. 493–498.

M. Brand and A. Hertzmann. 2000. Style Machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, 183–192.

C. Chiu and S. Marsella. 2011a. A style controller for generating virtual human behaviors. In *The th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1023–1030.

C. Chiu and S. Marsella. 2011b. How to Train Your Avatar: A Data Driven Approach to Gesture Generation. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents*. Springer, 127–140.

C. Chiu and S. Marsella. 2011c. How to Train Your Avatar: A Data Driven Approach to Gesture Generation. In *Proceedings of the 10th international conference on Intelligent Virtual Agents*. 127–140.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:cs.CL/1406.1078

L. Crnkovic-Friis and L. Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. *CoRR* abs/1605.06921 (May 2016).

N. Degara, E. A. Rua, A. Pena, S. Torres-Guijarro, M. E. Davies, and M. D. Plumbley. 2012. Reliability-Informed Beat Tracking of Musical Signals. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (Jan. 2012), 290–301.

Y. Ding, M. Radenen, T. Artieres, and C. Pelachaud. 2013. Speech-driven eyebrow motion synthesis with contextual Markovian models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3756–3760.

S. Dixon. 2006. Onset Detection Revisited. In *Proceedings of the 9th Int. Conference on Digital Audio Effects (DAFx'06)*, Vol. 120. 133–137.

C. Donahue, Z. C. Lipton, and J. McAuley. 2017. Dance Dance Convolution. (March 2017). arXiv:1703.06891

J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. (April 2017). arXiv:1704.01279

J. Ens and P. Pasquier. 2020. MMM : Exploring Conditional Multi-Track Music Generation with the Transformer. arXiv:cs.SD/2008.06048

K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. 2015. Recurrent Network Models for Human Dynamics. (Aug. 2015). arXiv:1508.00271 https://arxiv.org/abs/1508.00271

A. Gopalakrishnan, A. Mali, D. Kifer, L. Giles, and A. G. Ororbia. 2019. A Neural Temporal Model for Human Motion Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12116–12125.

F. S. Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48.

A. Graves. 2013. Generating Sequences With Recurrent Neural Networks. (Aug. 2013). arXiv:1308.0850

E. Herrmann, M. Manns, H. Du, S. Hosseini, and K. Fischer. 2017. Accelerating statistical human motion synthesis using space partitioning data structures: ccelerating statistical human motion synthesis. *Computer Animation and Virtual Worlds* 28, 6 (2017), e1780.

D. Herzog, V. Krueger, and D. Grest. 2008. Parametric Hidden Markov Models for Recognition and Synthesis of Movements. In *Proceedings of the British Machine Vision Conference*. 163–172.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 9 (Nov. 1997), 1735–1780.

G. Hofer. 2009. *Speech-driven animation using multi-modal hidden Markov models.* PhD Dissertation. University of Edimburgh.

D. Holden, T. Komura, and J. Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Transaction on Graphics* 36, 4, Article 42 (July 2017), 13 pages.

D. Holden, J. Saito, and T. Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics* 35, 4, Article 138 (July 2016), 11 pages.

A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. 2015. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. *CoRR* abs/1511.05298 (2015). arXiv:1511.05298 http://arxiv.org/abs/1511.05298

D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:cs.LG/1412.6980

J. Lee, S. Kim, and K. Lee. 2018. Listen to Dance: Music-driven choreography generation using Autoregressive Encoder-Decoder Network. *arXiv:1811.00818 [cs]* (Nov. 2018). http://arxiv.org/abs/1811.00818 arXiv: 1811.00818.

S. Levine, P. Krähenbühl, S. Thrun, and V. Koltun. 2010. Gesture controllers. *ACM Transactions on Graphics* 29, 4, Article 124 (jul 2010), 11 pages.

S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun. 2012. Continuous Character Control with Low-dimensional Embeddings. *ACM Transaction on Graphics* 31, 4, Article 28 (July 2012), 10 pages.

C. Li, Z. Zhang, W. Sun Lee, and G. Hee Lee. 2018. Convolutional Sequence to Sequence Model for Human Dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5226–5234.

Z. Li, Y. Zhou, S. Xiao, C. He, Z. Huang, and H. Li. 2017. Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis. *arXiv:1707.05363 [cs]* (July 2017). http://arxiv.org/abs/1707.05363 arXiv: 1707.05363.

J. Martinez, M. J. Black, and J. Romero. 2017. On human motion prediction using recurrent neural networks. (May 2017). arXiv:1705.02445

F. Ofli, Y. Demir, Y. Yemez, E. Erzin, A. M. Tekalp, K. Balcı, İ. Kızoğlu, L. Akarun, C. Canton-Ferrer, J. Tilmanne, et al. 2008. An audio-driven dancing avatar. *Journal on Multimodal User Interfaces* 2, 2 (sep 2008), 93–103.

F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp. 2012. Learn2Dance: Learning Statistical Music-to-Dance Mappings for Choreography Synthesis. *IEEE Transactions on Multimedia* 14, 3 (June 2012), 747–759.

D. Pavllo, D. Grangier, and M. Auli. 2018. QuaterNet: A Quaternion-based Recurrent Model for Human Motion. *arXiv:1805.06485 [cs]* (May 2018). http://arxiv.org/abs/1805.06485 arXiv: 1805.06485.

G. Peeters. 2004a. *A large set of audio features for sound description (similarity and classification) in the CUIDADO project.* Technical Report. Icram.

G. Peeters. 2004b. *A large set of audio features for sound description (similarity and classification) in the CUIDADO project.* Technical Report. Icram.

M. Pettee, C. Shimmin, D. Duhaime, and I. Vidrin. 2019. Beyond Imitation: Generative and Variational Choreography via Machine Learning. *arXiv:1907.05297 [cs, stat]* (July 2019). http://arxiv.org/abs/1907.05297 arXiv: 1907.05297.

S. Ross, G. Gordon, and D. Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*. PMLR, 627–635.

A.-A. Samadani, E. Kubica, R. Gorbet, and D. Kulić. 2013. Perception and Generation of Affective Hand Movements. *International Journal of Social Robotics* 5, 1 (2013), 35–51.

S. Starke, H. Zhang, T. Komura, and J. Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics* 38, 6, Article 209 (Nov. 2019), 14 pages.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems* 27 (2014), 3104–3112.

H. Tachibana, K. Uenoyama, and S. Aihara. 2017. Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention. *arXiv:1710.08969 [cs, eess]* (Oct. 2017). http://arxiv.org/abs/1710.08969 arXiv: 1710.08969.

N. Taubert, A. Christensen, D. Endres, and M. A. Giese. 2012. Online simulation of emotional interactive behaviors with hierarchical Gaussian process dynamical models. In *Proceedings of the ACM Symposium on Applied Perception*. ACM, 25–32.

G. W. Taylor. 2009. *Composable, distributed-state models for high-dimensional time series.* Ph.D. Dissertation. University of Toronto.

G. W. Taylor and G. E. Hinton. 2009. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, 1025–1032.

J. Tilmanne and T. Dutoit. 2010. Expressive gait synthesis using PCA and Gaussian modeling. In *Proceedings of the Third international conference on Motion in games*. Springer-Verlag, 363–374.

J. Tilmanne, A. Moinet, and T. Dutoit. 2012. Stylistic Gait Synthesis Based on Hidden Markov Models. *EURASIP Journal on Advances in Signal Processing* 2012, 1 (2012), 72.

J. M. Wang, D. J. Fleet, and A. Hertzmann. 2007. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 975–982.

Y. Wang, Z.-Q. Liu, and L.-Z. Zhou. 2005. Learning hierarchical non-parametric hidden Markov model of human motion. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*. IEEE, 3315–3320.

Z. Wang, J. Chai, and S. Xia. 2019. Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2019), 14–28.

Chapter 9

Summary and Conclusion

9.1 Summary of the Contributions

The research presented in this thesis aims to build machine-learning-based solutions for automatic movement generation. To this end, we formulate four research questions and present our work addressing each. In response to the first research question (**RQ 1**), our contribution is an extensive review of state of the art on training using machine learning models on motion capture data to build generative movement models. In response to **RQ 2**, we contribute to the human movement computing community by providing two high-quality motion capture datasets along with tools and libraries that we have developed to support the research on motion capture data. Our remaining contributions are on movement generation models. Following **RQ 3**, our third contribution is a neural-network-based walking movement controller that allows controlling the movement along multiple dimensions of planning, expression, and personal movement signature. Finally, in response to **RQ 4**, we present a model for generating dance movements given a music track in real-time. In the following, we revisit each contribution and identify the limitations of our approaches and possible future work.

9.1.1 Review of the State of the Art

We started our research by compiling an extensive review of state of the art on training machine learning models on motion capture data to create automatic movement generation systems to our first research question (**RQ 1**). In Chapter 2, we note that believability, control and manipulation, and real-time and interactive generation are the common goals and directions that are pursued in the literature (Section 2.2). We propose a multi-layer framework to characterize the movements of virtual agents (Figure 2.3). The purpose of this framework is to disambiguate the multifaceted nature of movement – the fact that multiple factors at different dimensions of function, planning, expression, and execution influence movement. We then use this framework to classify the reviewed works in terms of which dimensions of movement, if any, they model and control (Section 2.3). We review the publicly available movement datasets and discuss different movement representation and pre-processing algorithms used in the literature to train machine learning models on motion capture data (Section 2.4). We summarize, discuss, and analyze the machine learning models used to learn and generate movement, identifying each model's key advantages and limitations and outlining the possible solutions (Section 2.5). We highlight the lack of a rigorous and consistent evaluation framework for movement generation models among the reviewed works, which hinders the evaluation and comparative study of the proposed works in the literature (Section 2.6). Finally, we summarize our findings based on our review of the papers (Section 2.7).

9.1.2 Data and Tools

Throughout our research on movement generation, we needed movement data, tools, and libraries to explore, implement, and evaluate our ideas, which resulted in the formulation of our second research question (**RQ 2**). The outcomes of our efforts in addressing **RQ 2** are the contributions we summarize in the following.

To conduct research on affect-expressive movement generation, there were no datasets that contained variations of affect expression in movement based on a dimensional representation of affect (as opposed to a categorical representation). Similarly, to research on music-driven dance generation model, there were no datasets that contained synchronized dance and music data. Consequently, we curated, captured, and published the *Affect-Expressive Movements* and the *GrooveDB* datasets. In addition, to provide easy access our data in different formats, we present the MoDa 2.0 web-based frontend. MoDa 2.0 not only provides access to the data collected for the research presented in this thesis, it is also home to the data that our research group has collected (Section 3.2).

Working with movement data, we noticed that it is not always easy to analyze the content of the data or evaluate a particular movement feature extraction algorithm. As a result, we developed Mova, an interactive, online movement analytics platform. As presented in Chapter 4, Mova allows drawing the poses in the movement along with the visualization of multi-modal data and extracted features in parallel. Using information visualization techniques, Mova helps users to understand the contents of the data better. Users can use Mova to analyze, extract, and represent human movement data in terms of spatial, temporal, and qualitative characteristics.

To support building machine learning models using modern machine learning libraries and integrating the animation and analysis of movement data within the commonly used machine learning pipelines, we developed an open-source Python library called PyMO. Introduced in Section 3.5, PyMO provides the implementation of operations on motion capture data for machine learning applications and tools to visualize and animate movements that one can easily integrate into Jupyter Notebooks for easier handling of the research process.

We also contributed to the m+m software framework by integrating Mova and MoDa into the framework. The m+m framework facilitates developing interactive systems that use movement data, as presented in Appendix E. By integrating Mova and MoDa, we allow users to use the data from MoDa directly in their systems and use Mova to analyze the data and movement features.

Following our research group's work on affective computing on soundscapes (Thorogood and Pasquier, 2013; Fan et al., 2015) and music (Fan et al., 2017; Kıvanç Tatar, 2017), we investigated affect recognition from human movement. Our research group conducted a study that shows that unbiased human observers can perceive the affective state of a moving animated avatar from its movements to a reasonable extent (Li and Pasquier, 2016; Li et al.,

2018). Building on this finding, we developed a machine learning model for estimating the perceived affect levels of full-body motion capture data using a pair-wise ranking approach. We present this work in Appendix D.

Limitations and Future Work

We advocate for more diverse datasets and more comprehensive tools and libraries within the field of human movement computing.

More diverse movement datasets are essential for the progress of the field of human movement computing. By diversity, we refer to both the diversity in the repertoire of the movements and the diversity of the human subjects being recorded. Humans are capable of performing a large repertoire of movements. Datasets that include a more diverse repertoire of movements open the way for a wider range of applications for movement computing. Such datasets also can be used to build more robust computational models of movement. It is also essential to include a more diverse range of human subjects when recording the data. Movements of humans are as diverse as humans. How each individual moves is influenced by elements such as gender, cultural backgrounds, physical build, and socioeconomic status (Bartenieff and Lewis, 1980; Serlin et al., 2007; Caldwell, 2013). It is therefore essential to capture and model the movements of a diverse group of humans to create computational models of movement that are less biased towards specific groups or cultures and more closely correspond to the representation of humans in the real world.

Since we started our research, a growing number of public motion capture datasets are being provided by universities and research groups. However, as noted in Section 2.8, there are still large gaps in the diversity of the content of such datasets. While there is a large amount of motion capture data actively recorded, most of these data are proprietary and are not available to the broader research community. We believe more collaborations between universities and the industry can promote the release of more public industry-quality and diverse datasets. Recent datasets such as the Ubisoft La Forge Animation Dataset(Harvey et al., 2020) are encouraging examples of publically available industry data.

Unlike computer vision, natural language processing, and music, the field of human movement computing lacks comprehensive tools and libraries that provide extensive sets of feature extraction algorithms, tests, metrics, and evaluation frameworks. Libraries such as OpenCV[1] for computer vision, Natural Language Toolkit (NLTK)[2] for natural language processing, and Essentia[3] for audio and music applications have facilitated the works of not only researchers but also software developers and artists. The lack of such libraries

[1] `https://opencv.org`

[2] `https://www.nltk.org`

[3] `https://essentia.upf.edu`

for human movement computing hinders the field's progress and its adaptation by a wider community of researchers, software developers, and artists. We hope that our contributions outlined here are the first steps in this direction, and we advocate for a collective effort by the community to build such a comprehensive framework as a significant direction for future work.

9.1.3 Movement Generation and Control

To address our third research question (**RQ 3**) on style imitation in movement generation, we take a data-driven and machine learning approach. We contribute to **RQ 3** in two iterations. First, we present a model called AffectNet to generate movements while controlling the mover's affective states. Next, we improve this model by integrating the control of multiple dimensions of movement (Section 2.3.2), namely the expressive, the planning, and the personal movement signature, into a single walking movement generation model called WalkNet. With these two models, we show that we can build movement controllers using neural networks. Such controllers not only can generate movement based on the given description of factors from multiple movement dimensions, but they also can generate movement for variations that did not exist in the training data.

Limitations and Future Work

We recognize that one of the limitations of our approach to capturing the affective qualities of movement is the lack of diversity in our training dataset. As we mentioned in Section 9.1.2, it is essential to capture the movements of a diverse group of human subjects to avoid biasing the models towards a specific group of people. The need for such diversity becomes more apparent in the field of affective computing where personal and cultural variations play an important role in how the internal states of movers are exhibited in and interpreted from their movements . However, our dataset at the time when we conducted our research only included data from two human subjects from similar backgrounds, which biases our models towards a very small group of people. Future larger-scale studies have to be done to capture and model affective qualities of movement based on a more representative group of people.

Since publishing our paper on WalkNet, we have added ten more performers to our Affect-Expressive Movements dataset. One direction of future work is to train WalkNet on this larger, 12-performer dataset. We can then experiment with learning latent representations of the personal movement signature.

Another limitation of our movement generation models is the lack of constraints on the end-effectors. Enforcing such constraints allow us to model the interactions of animated agents with their environment and external objects. For example, picking up an object from the table with the right hand requires ensuring that the hand's position matches the object's position. Such constraints can also help limit unwanted artifacts in the generated data, most notably the foot sliding.

Our approach to movement generation and control is purely data-driven. While data-driven methods are good at capturing and generating the expressive qualities of movement, they are, to a high degree, limited to the contents of the datasets they are trained on. Moreover, data-driven methods do not explicitly adhere to the constraints and the laws of physics. Although data-driven methods can be trained to approximate such constraints (e.g., (Starke et al., 2019)), they require an extensive training set.

Another group of movement generation approaches uses physical simulations in order to model and generate movement animation (Safonova et al., 2004; Agrawal et al., 2013). Incorporating physical laws allows such models to create physically valid movements, are proportionate to the body's physical dimensions, and react to the other forces in the environment such as gravity, friction, and external push or pull. However, purely physics-based methods are insufficient in modeling the expressive variations of movements.

Hybrid approaches that combine both data-driven and physics-based generation techniques can generate movements that are natural-looking and believable but also adapt to the changes in the environment and the specifications of the tasks the movement is performing (Peng et al., 2018). As a result, we are interested in integrating the affect-expressive movement controller we propose in this thesis with physics-based models in order to provide greater flexibility for controlling the movements of an agent.

9.1.4 Music-Driven Dance Generation

In AffectNet and WalkNet, we control the movement generation using low-dimensional stimuli (e.g., two dimensions for affect, two dimensions for direction, and two or more dimensions for the personal movement signature), which are highly correlated with the movement itself. In addition, compared to the framerate of the movement data, such stimuli are temporally course. To make a step further, through our fourth research question (**RQ 4**), we ask if we can control the movement generation using a more complex stimulus such as raw audio. Audio signals are temporally denser than the movement data (e.g., 44100 samples-per-second versus 30 frames-per-second) and follow a more complex mapping to movement.

In particular, dancing to music is a complex behaviour that constitutes highly non-linear and often subjective mappings between two modalities. To address **RQ 4**, we started working on a machine learning model called GrooveNet that generates grooving dancing movements based on a given audio track. We present our work in two iterations. In our first attempt to build GrooveNet, we used the same FCRBM architecture as for WalkNet. We created a small training dataset of synchronized music and motion capture data by recording a single dancer on three computer-generated electronic dance music songs. Before training the model, we extracted a set of audio features from the raw audio signals, which we fed as the stimuli to the context unit of the FCRBM. The results show that the model can generate simple dancing patterns when given songs from the training set. However,

the model fails to generate plausible human-like movements when given songs beyond the training set.

In the second iteration of GrooveNet, we recorded more training data from nine more dancers. We also used an RNN instead of FCRBM for its higher learning capacity. Our results show that when trained on the same data as the FCRBM, the RNN is better at generating plausible dancing movements even when given songs that are not in the training set. Overall, with GrooveNet, we show that it is possible to control the movement generation using multi-modal, complex stimuli such as raw audio.

Limitations and Future Work

One of the limitations of GrooveNet 2.0 is that it does not perform well when trained on the data from all the ten performers, compared to when trained on one or two performers. We plan on addressing this problem by adding a mixture layer as the output of the model. We are also interested in investigating ways to learn a representation of the dancer's style in such a way that we can control and manipulate the style during the generation.

We plan to further develop and fine-tune the GrooveNet model to generate a more extensive and natural repertoire of dance moves. We are particularly interested in exploring different machine learning techniques, such as incorporating an attention mechanism and using a differentiable forward kinematics loss function similar to the one used by Pavllo et al. (2018).

We also plan on using GrooveNet in interactive art installations. We are interested in exploring generating audio-visual elements based on the music provided by the audience. We are also interested in exploring the use of GrooveNet in applications where embedding the dynamics of movement in response to music can enhance the interactions with humans. For example, in designing products with moving parts such as the *R* dancing speaker from Teenage Engineering (Teenage Engineering, 2020), which moves its robotic arm based on the music that it is playing.

9.2 Final Discussion

In the following, we provide the final discussion and the future of our research on movement generation beyond the works presented in this thesis. In particular, we reflect on doing research in the field of machine learning, which has been seeing an unprecedented number of new techniques and approaches in a fast-changing landscape. We also highlight two directions that we plan on continuing our work on human movement computing.

9.2.1 Doing Research in a Fast-Changing Field

As we started our research on machine learning and movement generation, the field of machine learning was starting to undergo radical and fast changes that continue to the time

of this writing. On the one hand, newer machine learning techniques, faster hardware, new computing paradigms appear every few months. On the other hand, our research process was slow and often interrupted. While our research greatly benefited from the advances in the field, it was also impossible to keep up with all the new and relevant works.

In the beginning, the field was almost dominated by generative systems based on Hidden Markov Models (HMMs) following their dominance in speech recognition and synthesis. Gradually, and with the success of neural networks, models based on Restricted Boltzmann Machines (RBMs) showed promising and better results than HMMs. The next change came with the re-introduction of RNNs into sequence modelling. LSTMs and GRUs worked well with language, speech, and handwriting modelling and to some extend in movement generation. Although RNNs are powerful models for modelling sequences, they suffer from problems in modelling long sequences and training them to generate sequences of real-valued data such as motion capture.

More recently, a group of approaches propose a different paradigm in using neural networks for movement generation. Instead of recurrent or convolutional networks, a simple, slim feed-forward regression neural network takes the previous motion capture frame and a set of control parameters at each frame. The difference from the previous approaches is that for each frame, a function dynamically generates the weights of the regression neural network based on the phase information of the movement. As a result, the task of learning the dynamics of movement is moved from the feed-forward neural network to the weight generation function. There are different ways that one can build the weight generation function, such as a cubic Catmull-Rom spline function or another neural network. The resulting models have small computational and memory footprints. Also, the experiments show that the models are effective in constraining the generated movements based on the given control parameters (Starke et al., 2019, 2020).

9.2.2 Evaluation of Generative and Creative Systems

As discussed in Section 2.7.9, one of the challenging areas in the research on movement generation models is the evaluation of the system. The difficulties in evaluating any possible solution also influence the research process. As the research process is iterative, it requires constant evaluation, debugging, and refinement of the approaches. One of the sources of this challenge is due to the creative nature of human movement, where often there is no single "correct" way of performing a movement. Furthermore, it is challenging to quantify the quality, expressivity, and naturalness of the generated movements, which can facilitate the training and evaluation of machine learning models. In this context, we believe further research is needed to design and develop robust evaluation strategies. This includes identifying best practices, as well as defining quantitative metrics that better correspond to how humans evaluate the quality of the movements of an animated character.

One of the avenues of research is to use the features of neural networks that are pre-trained on large datasets to extract statistics that can indicate how likely it is for a given sample to belong to the distribution of real-world data. For example, in the field of computer vision, metrics such as the Inception Score (Salimans et al., 2016) and the Frećhet Inception Distance (FID) (Heusel et al., 2017) follow this strategy and are shown to correlate with how humans evaluate how realistic-looking generated images are for the type of images within a specific image dataset.

Another avenue of research is to investigate how human experts observe and evaluate human movement. Human experts in different domains such as sports, dance, and healthcare are trained to detect and describe various qualities of movement in their domain. Therefore, by gathering insights from their approaches, one can design better quantitative and qualitative evaluation methods for generative systems.

9.2.3 Exploring Movement Generation for Non-Humanoid and Physical Devices

Research has shown that the motion of abstract objects can communicate information such as intentions or emotions (Heider and Simmel, 1944; Bartram and Yao, 2008). Further works have proposed computational models for generating motion of abstract objects based on expressive descriptions of movement (Lockyer et al., 2015, 2016; Feng et al., 2017). The models we presented in this thesis so far are trained on human movement data and generate movements for animated humanoid characters. As a future work direction, we are interested in investigating applying human movement dynamics to movements of non-human-like objects, and thus bridging the fields of human movement computing and movement computing. Such objects can be virtual or physical and range from abstract visuals to automatic doors and robotic arms. To do so, we need to devise ways to re-target the movement dynamics that are learned from the movements of a human body to targets with different degrees-of-freedom and proportions.

9.2.4 Generative Models for Human Movement Computation

We are also pursuing using the movement-understanding that is learned by generative movement models for applications other than movement controllers for animation generation.

One of the topics that we are working on is mocap super-resolution, in which we use machine learning models to augment the data coming from a capture system with a low-degrees-of-freedom (DOFs) to a full-body motion capture rig. Our approach to this problem is to use generative models similar to those we introduced in this thesis. Since a generative model can learn the distribution of a subset of desired movements, it has a better chance of generating a plausible movement given a limited amount of data. We are pursuing this idea in two collaborative projects with industry partners. In one project, we partner with Inscape

Studios[4] to map movements of users wearing Virtual Reality (VR) headsets and using hand controllers and generate corresponding full-body poses of the player in real-time. In another project, we partner with Shocap Entertainment[5] and Animatrik Film Design[6] to develop a marker-based motion capture system that uses a reduced number of markers as opposed to the common marker sets and produce equivalently high-quality skeletal animation of the performer in real-time.

In another collaboration with Shocap Entertainment and Animatrik Film Design, we are working on developing models for movement recognition and tracking for live performance settings. The goal is to build models that recognize, in real-time, which of a number of pre-set 3D movements or gestures is or are being executed at a given time (if any). The model then indicates how far in the pre-set movement we are and how fast and in which direction the movement is being executed. The information from the model can then be used as cues for the audio and visual systems in the live performance. While these problems have been investigated for simple 2D or 3D gestures for relatively low dimensions (Bevilacqua et al., 2010; Caramiaux et al., 2014), mocap offers a more challenging context to which richer models need to be applied.

9.3 Conclusion

In the broad field of human movement computing, we focus on movement generation using data-driven machine learning techniques. We present a big picture of this field through a review of the works and methods for movement generation. We show that we can build generative models of movement that are able to control the movement based on a wide range of stimuli from planning parameters, affect-expression, and music. Finally, we publish the data, code, and libraries resulting from this research online and under free and open-source licenses. In an increasingly computerized world where we interact with various devices and technologies at home and work regularly, it is imperative to incorporate computational movement understanding within these devices and technologies. We look forward to extending our work in the future and seeing a stronger and richer field of human movement computing.

[4] https://www.inscapestudios.com/

[5] https://www.shocap.com/

[6] https://www.animatrik.com/

Bibliography

S. Agrawal, S. Shen, and M. van de Panne. 2013. Diverse motion variations for physics-based character animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium*. 37–44.

I. Bartenieff and D. Lewis. 1980. *Body Movement: Coping With the Environment*. Routledge.

L. Bartram and M. Yao. 2008. Animating Causal Overlays. *Computer Graphics Forum* 27, 3 (2008), 751–758.

F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana. 2010. Continuous Realtime Gesture Following and Recognition. In *Gesture in Embodied Communication and Human-Computer Interaction*. 73–84.

C. Caldwell. 2013. Diversity issues in movement observation and assessment. *American Journal of Dance Therapy* 35, 2 (2013), 183–200.

B. Caramiaux, N. Montecchio, A. Tanaka, and F. Bevilacqua. 2014. Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Transactions on Interactive Intelligent Systems* 4, 4, Article 18 (Dec. 2014), 34 pages.

J. Fan, K. Tatar, M. Thorogood, and P. Pasquier. 2017. Ranking-Based Experimental Music Emotion Recognition. *In Proceedings of the 18th International Society for Music Information Retrieval Conference*.

J. Fan, M. Thorogood, B. E. Riecke, and P. Pasquier. 2015. Automatic Recognition of Eventfulness and Pleasantness of Soundscape. In *Proceedings of the Audio Mostly 2015 on Interaction With Sound*. ACM, Article 12.

C. Feng, L. Bartram, and D. Gromala. 2017. Beyond Data: Abstract Motionscapes as Affective Visualization. *Leonardo* 50, 2 (2017), 205–206.

F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. 2020. Robust Motion In-Betweening. 39, 4 (2020).

F. Heider and M. Simmel. 1944. An Experimental Study of Apparent Behavior. *The American Journal of Psychology* 57, 2 (1944), 243–259.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 6629–6640.

P. P. Kıvanç Tatar. 2017. MASOM: A Musical Agent Architecture based on Self- Organizing Maps, Affective Computing, and Variable Markov Models. *In Proceedings of the 5th International Workshop on Musical Metacreation (MuMe 2017)*.

W. Li, O. Alemi, J. Fan, and P. Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In *Proceedings of the 5th International Conference on Movement and Computing (MOCO '18)*. ACM, Article 18.

W. Li and P. Pasquier. 2016. Automatic Affect Classification of Human Motion Capture Sequences in the Valence-Arousal Model. In *Proceedings of the 3rd International Symposium on Movement and Computing*. ACM, Article 15.

M. Lockyer, L. Bartram, T. Schiphorst, and K. Studd. 2015. Extending Computational Models of Abstract Motion with Movement Qualities *(MOCO '15)*. ACM, 92–99.

M. Lockyer, L. Bartram, T. Schiphorst, and K. Studd. 2016. Enhancing Visualization with Expressive Motion. *Electronic Imaging* 16 (2016), 1–6.

D. Pavllo, D. Grangier, and M. Auli. 2018. QuaterNet: A Quaternion-based Recurrent Model for Human Motion. *arXiv:1805.06485 [cs]* (May 2018). http://arxiv.org/abs/1805.06485 arXiv: 1805.06485.

X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics* 37, 4, Article 143 (July 2018), 14 pages.

A. Safonova, J. K. Hodgins, and N. S. Pollard. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '04)*. 514–521.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc.

I. A. Serlin, M. R. Berger, and R. Bar-Sinai. 2007. Moving Through Conflict: Understanding Personal and Cultural Differences Through Movement Style. *Journal of Humanistic Psychology* 47, 3 (2007), 367–375.

S. Starke, H. Zhang, T. Komura, and J. Saito. 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics* 38, 6, Article 209 (Nov. 2019), 14 pages.

S. Starke, Y. Zhao, T. Komura, and K. Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics* 39, 4, Article 54 (July 2020), 14 pages.

Teenage Engineering. 2020. R. Retrieved 2020-10-30 from https://teenage.engineering/designs/R

M. Thorogood and P. Pasquier. 2013. Impress: A Machine Learning Approach to Soundscape Affect Classification. In *Proceedings of the 13th International Conference on New Interfaces for Musical Expression*. 256–260.

Appendix A

SIAT Guidelines for Writing a Cumulative Thesis

SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

SIAT GUIDELINES
WRITING A CUMULATIVE THESIS
Produced by the SIAT Graduate Program Committee, 6 November 2013

This document is adjunct to SIAT's calendar entry and to SFU's Graduate General Regulations. It describes SIAT's normal practice with respect to the topic it addresses.
It has been approved by SIAT's Graduate Caucus.

The PhD thesis may have the form of a monograph (i.e., the classic thesis format of one single document), or of a compilation with a number of scholarly peer-reviewed articles ("cumulative thesis"). Students should always consult early on with their senior supervisor and committee, who will discuss with them and decide what form of thesis is the most suitable for a given case.

Note that these guidelines described here are only general guidelines, and the students' supervisory committee might have different requirements and will have the ultimate say in what contribution is required for a thesis. Master's thesis may also be in the form of a cumulative thesis if suitable and advised and approved by the senior supervisor(s).

1. In the case of a cumulative thesis, these scholarly articles are to be connected by an initial introduction chapter (explaining the subject and scope of the work and how the different articles contribute) and a summative final discussion chapter (that includes the overall contributions, main conclusions, and an outlook). This serves to interrelate the contributions as well as discuss and draw conclusions from the entire work. The publications need to be integrated as chapters into the theme of the thesis and must deal with the overall topic of the thesis. The thesis should have continuous pagination and an aggregated bibliography. The individual papers may still have their own bibliography in addition.

2. The thesis should have a content and depth corresponding to a classic (monograph-style) thesis. E.g., this might be achieved by about 3-6 peer-reviewed conference papers, journal articles, or other scholarly contributions of high value where the supervisory committee assesses the quality as being appropriate. The articles should maintain such a level that they could be accepted for publication in an international scholarly journal with a rigorous referee procedure. At least two of these articles should already have been accepted for publication or be published. All may already be published. For published articles, the comprehensive biblio graphic reference should be stated. For accepted or submitted manuscripts, the venue and date of acceptance/submission should be included.

3. The individual articles may have been written together with the main supervisor, another supervisor or other persons. In order to show that the candidate has attained the intended proficiency, the majority of the articles must have been written by the candidate personally (which should normally result in first authorship) and comprise a substantial part of the thesis as a whole. The thesis should be accompanied by a detailed descrip tion of all the authors' contributions to each of the articles.

4. In terms of its scholarly contribution, a cumulative thesis in its entirety shall satisfy the same academic requirements as a thesis in the form of a monograph. Any submission of constituent parts that had been published/submitted/written before the student started his/her doctoral/masters studies or is primarily based on prior work may be included to contextualize the thesis if properly declared, but typically does not count towards the scholarly contribution of the thesis.

5. Before your article is published by a publisher, you can attempt to retain specific rights to your work through a publication agreement addendum, such as the SPARC Canadian Author Addendum. If you did not retain rights to re-use your article, you can request permission to include your article in your thesis by emailing or writing to the copyright holder, explaining how and why you want to use the work and requesting permission. If granted permission, you should keep a record of who gave the permission, what was permitted, the date, and how to contact the person who gave the permission. A copy of each relevant copyright release must be included in your Thesis Package. If the publisher will not grant you permission, it may still be possible to use the content of the pre-print or post-print of your article, depending on the publisher's copyright policies outlined in the publisher's copyright transfer / author publication agreement, many of which can be found on the SHERPA/RoMEO website. You can find the latest information about copyright on the Copyright at SFU website. If you have questions about retaining or obtaining copyright permission, you can always contact your liaison librarian.

For reference: Original regulations
http://students.sfu.ca/calendar/interactive-arts-technology/iat-PhD.html#program

Appendix B

Supplemental Material: Source Codes

Description: The accompanying ZIP file contains the source code for Mova, AffectNet, WalkNet, GrooveNet 1.0, and GrooveNet 2.0.

Filename: `oalemi_phdthesis_sourcecode.zip`

URL: `https://omid.al/phdthesis`

Appendix C

Supplemental Material: Outputs from AffectNet, WalkNet, and GrooveNet

Description: The accompanying material contains video files created from sample outputs from AffectNet, WalkNet, and GrooveNet models.

Filename: `oalemi_phdthesis_outputs.zip`

URL: `https://omid.al/phdthesis`

Appendix D

Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space

This appendix presents content from the following paper:

William Li, Omid Alemi, Jianyu Fan, and Philippe Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In Proceedings of the 5th International Conference on Movement and Computing (MOCO '18). ACM, Article 18, 8 pages. DOI: https://doi.org/10.1145/3212721.3212813

Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space

William Li
Simon Fraser University
dla135@sfu.ca

Omid Alemi
Simon Fraser University
oalemi@sfu.ca

Jianyu Fan
Simon Fraser University
jianyuf@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

ABSTRACT

Affect estimation consists of building a predictive model of the perceived affect given stimuli. In this study, we are looking at the perceived affect in full-body motion capture data of various movements. There are two parts to this study. In the first part, we conduct groundtruthing on affective labels of motion capture sequences by hosting a survey on a crowdsourcing platform where participants from all over the world ranked the relative valence and arousal of one motion capture sequences to another. In the second part, we present our experiments with training a machine learning model for pairwise ranking of motion capture data using RankNet. Our analysis shows a reasonable strength in the inter-rater agreement between the participants. The evaluation of the RankNet demonstrates that it can learn to rank the motion capture data, with higher confidence in the arousal dimension compared to the valence dimension.

CCS CONCEPTS

• **Computing methodologies** → *Modeling methodologies*;

KEYWORDS

Affective computing, motion capture, machine learning

ACM Reference Format:
William Li, Omid Alemi, Jianyu Fan, and Philippe Pasquier. 2018. Ranking-Based Affect Estimation of Motion Capture Data in the Valence-Arousal Space. In *MOCO: 5th International Conference on Movement and Computing, June 28–30, 2018, Genoa, Italy.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3212721.3212813

1 INTRODUCTION

In the recent growing interest of developing technology to recognize people's affective states [15], more and more studies have shown that body expressions are effective in conveying emotion [3, 40]. Combined with the increase in the volume of sheer amounts of data, there is an increasing demand for the development of affect recognition systems which in turn has potential impacts in clinical and entertainment contexts. Thus, we developed an affect ranking system using the valence-arousal (VA) model of human emotion, and used full body motion capture data as input, which does not contain any information regarding facial expressions or voice. When considering the three aspects of movement, functional (the task of the movement, such as picking up a cup), executional (the pattern of movement, such as using the left or right hand to pick up the cup), and expressive dimensions (the emotions behind the movement) [1], we are essentially measuring the expressive dimension of full body movements.

The ground-truthing experiment was conducted using a ranking system in which we ask participants to rank the relative valence and arousal for different pairs of movements. This results in a complete relative ranking of the movements. The machine learning models are trained on this ranking.

The contribution of this paper is a first step in the processing and affect estimation of a large amount of motion capture data, leading to future off-line and on-line applications. The main off-line applications involve database labelling, which is especially useful for the development of movement databases [32]. A valid and reliable ranking model for movement expressivity would allow us to automatically label existing motion capture data according to the valence-arousal model. In on-line scenarios, such a model could be used in interactive arts or therapy contexts. Such a system can also be used in generating movement with user-specified valence and arousal [1]. Our goal is therefore to estimate affect expressed by movement using the VA model, specifically a meaningful rank on the VA spectrum relative to the other data.

For the rest of the paper, we start by outlining the related work in affect classification. After that, we describe the data and the processing used in our study, followed by experimental methods, participants, results, and analysis for each iteration separately. Lastly, we end with concluding remarks and future work.

2 PREVIOUS WORK

2.1 Affect Estimation

In affect estimation, considerations that come into play include the intended emotion of the mover, and the perceived emotion of the mover [24, 29]. Malandrakis et al. [27] have shown that there can be a difference even in award-winning movies in the intended and experienced emotions. How well the intended emotions are portrayed plays an important role in the movies. With their experiment, they used award-winning films and expert annotators to narrow the gap between the intended and expressed emotions. In our experiment, we are able to direct the movers, so we assume the intended affect is identical to the perceived affect.

MOCO, June 28–30, 2018, Genoa, Italy

ACM ISBN 978-1-4503-6504-8/18/06…$15.00
https://doi.org/10.1145/3212721.3212813

In the field of affective computing, facial expressions are often examined in the determination of affective states [13, 19]. However, Inderbitzin et al. [21] have shown that it is possible to perceive VA states from movement even on faceless generated characters, regardless of viewing angle. They have even identified some canonical parameters that control the expression of emotions in locomotive behavior, such as upright upper body postures being perceived as more emotionally positive and vice-versa for forward leaning postures. Other documented sources also suggest that humans convey emotions through body movement and postures [10, 11]. Analysis of head pose and movement is able to achieve 71.2% accuracy in recognizing depression [2]. Furthermore, studies in movement have shown certain features in expressive movement, such as portrayal of strength, can be linked to specific emotions, such as fear or anger [11, 41].

In affect estimation based on body movement, there have been many studies in using dance with mixed results ranging from barely above random chance to close to human levels of accuracy [8, 22, 33]. Kapur et al. developed classifiers that achieved comparable accuracy as observers using dance movements [23]. However, as Kleinsmith points out, dance is often exaggerated to convey affect [24].

Looking at non-dance-based systems, Castellano et al. have attempted to infer emotional states using video analysis on movement qualities such as amplitude, speed, and fluidity. Their system was able to discriminate between "high" and "low" arousal emotions and "positive" and "negative" [9]. Pollick et al. conducted a study to compare the performance of their automatic system with human recognition. In their study, they used 3D positioning measurements of the arm in knocking, lifting, and waving motions with two affective states, neutral and angry. They concluded that the automatic system was able to discriminate between the two states more consistently than humans [34]. Samadani et al. developed a system for both full body as well as hand-arm improvisiation movements to discriminate between 4 affective states using HMMs with good results [38].

Nicolau et al. developed a system for estimation of affect modalities in the Valence-Arousal space using multi-modal inputs (based on facial expression, shoulder gesture, and audio cues). Their approach claims to be unique in that it performs *continuous* affect prediction according to the valence-arousal model. They compare both Support Vector Machines (SVM) and bi-directional Long-Short-Term Memory Neural Networks (BLSTM-NN), concluding that BLSTM-NN performs better [31]. However, we have decided not to use BLSTM-NN due to the fact that they are using different sets of input data (extracting data from video and audio as well as mainly focusing on facial expressions); in our case we are using motion capture data with no facial expressions. Furthermore, the lack of a benchmark and standard skeleton markers due to the use of different datasets and body markers in the aforementioned studies makes it difficult to compare and evaluate different systems.

2.2 Ranking and Rating

Most of the previous research have used a rating system. However, Yannakakis et al. point out some limitations to using ratings in 2015 [42]. Firstly, inter-personal differences including cultural background and experiences can lead to different perceptions of affects. What appears to be happy to one person might appear neutral to others. Similarly, Baveye et al. in 2014 [4] point out that using ratings require the participant to understand the full range of the valence and arousal scale of the data, which is usually not feasible. Secondly, Yannakakis et al. argue that using adjectives such as "moderately" and "extremely" are not numbers, and thus any method that treats them as numbers such as average values or t-tests are fundamentally flawed. This also ties in with the third issue they point out that ratings are not always linear.

Therefore, we have chosen to use a ranking system for classification. Ranking approaches are easier in terms of cognitive load and have a higher inter-rater agreement [4]. Yannakakis et al. [42] also claim that using rankings eliminates the cultural and subjective biases in the annotation. However, a disadvantage to using rankings is that the ranks do not indicate the distance between them. For example, we know video A has higher valence than video B, but the amount by which it is higher is not clear.

There are three approaches to the learning to rank: pointwise, pairwise, and listwise [25]. In the pointwise approach, the model predicts a score for a single input item. In the pairwise approach, the model is given two input items, and ranks them accordingly. In the listwise approach, a set of items is given to the model, in which it outputs a ranked list of the input items. As in a similar experiment by Fan et al. [12], we will be going with a pairwise approach, as it produced the best results.

There are a number of systems used in the literature for pairwise ranking, including SortNet [36], RankNet [7], RankSVM [20], and RankBoost [16]. We use RankNet as it is an efficient model for working for high-dimensional motion capture data.

2.3 Model of Affect

For our study, we will be using Russell's model of affect [37]. A potential drawback of Russell's model is that some researchers such as Fontaine et al. [14] are starting to believe more dimensions are needed to describe the emotional space, such as the PAD (Pleasure-Arousal-Dominance) model put forth by Mehrabian et al. [28], which includes the addition of the dominance dimension. Dominance refers to whether an affect is controlling, submissive, or otherwise influenced by something else. For example, if a person is subjecting themselves to the command of another person or feel pressured or otherwise controlled by another entity, this would be submissive on the dominance dimension. We did not include the dominance dimension in our experiments because all of our movement are performed individually without another body or objects in the scene. Thus the concept of dominance does not make sense in the context of our motion capture data.

Another potential drawback is that Schacter et al. [39] claimed that the physiological reactions contribute to the emotional experience by facilitating a cognitive recognition of a physiologically stimulating event, which then defines the emotional experience. The emotion is the result of a combination of the cognition of a physiological event and the participants' reception of adrenaline. For our studies, we do not address nor do we have control over either the actors' or viewers' physiological states. With our actors, we are assuming that they are able to act the specified emotional state despite their internal physiological states. With the viewers, we are assuming their physiological states do not drastically affect their perception of the movements while watching the animation as it is a low-stimulus activity.

Other than dominance, there is not yet a well-established set of dimensions in addition to valence and arousal that are considered necessary in order to describe affect. Determining such a set of

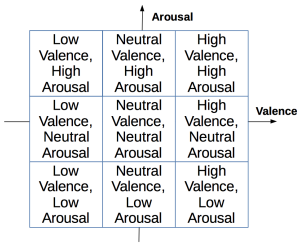


Figure 1: Valence and arousal combinations

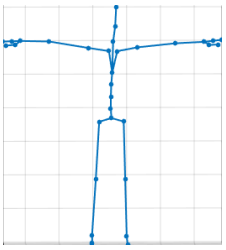


Figure 2: MoCap skeleton

standards is outside the scope of this study. Therefore, we will be using just valence and arousal as the basis for our model of affect, similar to other previously cited studies. However, we will keep in mind that as research in the dimensions of affect progresses, our experiments will potentially need to be replicated or our models tweaked to account for additional dimensions or new models of affect. To our knowledge, automatic systems in affect estimation using motion capture data has not yet been attempted in the context of a dimensional model of affect.

3 DATASET

3.1 Motion Capture Data

As part of the efforts of the MovingStories project, an open source MoCap database[1] [32] has been created. For this study, we are using some of the recordings in this database that have been labelled according to the circumplex model of affect [37]. The data are in the form of MoCap bvh files and accessible in the MoCap database[2]. The key point for this experiment is that only the skeletal information is retained in the end. There are no facial expressions and nothing that explicitly indicate gender, body type, or ethnicity. Three professional actors, one male and two female, performed in the data collection stage. Two of the actors performed 9 different types of movements: walking in a figure eight pattern, hugging, static improvisation, free improvisation, sitting down, pointing while sitting, walking with sharp turns, improvisation while facing another actor, and lying down. Improvisation refers to a movement that is at the discretion of the mover. In other words, they are given the direction of acting in a certain affect, but are free to carry out whatever movement they wish that they believe would illustrate that affect. The difference between free and static improvisation is that in static improvisation the only additional restriction is that they must remain standing in the same spot. The third mover performed only the two walking movements. There are 9 takes for each movement, corresponding to the 9 different possible VA combinations shown in Figure 1, covering more emotional states than similar existing datasets (e.g. 4 emotions in the library presented by Ma et al. [26]). Existing labels were created by dividing the Russell's model [37] into low, neutral, and high along both the valence and arousal axis. Using this model, anger would be classified as low on the valence axis but high on the arousal axis.

[1] http://moda.movingstories.ca/
[2] http://moda.movingstories.ca/projects/29-affective-motion-graph

The data were recorded with a Vicon motion capture system[3] and mapped to a skeleton representation with 30 joints as shown in Figure 2. Eighteen of the motion capture files were recorded at 60 frames-per-second, while the rest were recorded at 120 frames-per-second. Therefore, we downsample 120fps files to 60fps. Sequences vary in length from 1000 frames to 8000 frames. Each frame contains the rotations for each of the joints in the Euler representation, as well as the spatial location and orientation of the skeleton root.

3.2 Pre-Processing

We convert the rotational data from Euler representation to the exponential maps [18] as suggested in the literature for training neural networks on motion capture data. After removing empty dimensions as well as orientation and translation of the skeleton's root to eliminate bias due to geometrical translation, we are left with a 46-dimensional vector per frame. We further concatenate a window of consecutive frames into one feature vector to flatten the time dimension. We experiment with window sizes of 1, 3, and 12 frames. The intent behind using these window sizes is to test a variety of window sizes and determine if there is a trend in the performance of the model relative to the window size. However, we acknowledge that at these window sizes, the duration captured is very short. It is closer to a snapshot of the movement rather than fully capturing the temporal aspect of the movement. Based on previous works, it is not clear at this time what sort of high-level features would be useful in recognizing affect in full-body motion capture data that would persist over a long period of time. Depending on the models, long window periods can also significantly increase processing time, which would be detrimental to most practical applications. Hence we test using small window sizes. Converting to real time using the frame rates, these window sizes account for 0.1 second or less of the movements. Therefore, the window sizes are not realistic to the perception or reflex of humans. From the perspectives of a machine, these window sizes are enough to see a trend in the performance of the models. Furthermore, building systems using smaller window sizes will also be more advantageous in any application that relies on real-time affect estimation in order to give a prediction as fast as possible.

4 DATA COLLECTION

We conduct our survey on the CrowdFlower platform. Full documentation for this platform can be found at the CrowdFlower

[3] https://www.vicon.com/

website[4]. We chose this platform due to a similar successful study in affective rankings. Baveye et al. [5] conducted a study where 9800 videos were ranked on CrowdFlower in terms of valence and arousal. Their goal was to provide rankings for the affective video database LIRIS-ACCEDE. We obviously cannot use this database as our videos are motion capture data, but as our goals are similar, we have chosen to use the same experimental protocol that they have.

The CrowdFlower platform was chosen because it reaches several crowdsourcing services, which also allows for a good distribution of demographics [4]. However, the number of labor channels have been reduced in the last few years. It is no longer disclosed by CrowdFlower where they may distribute their tasks nor is it possible to choose specific channels. The advantages of using crowdsourcing platforms like CrowdFlower include supported infrastructure in both the survey design and payment. For example, CrowdFlower offers templates for a variety of surveys that are all fast and easy to implement. This reduces the potential problems that may come up from the researchers having to host their own online survey such as website or server issues. Another concern with gathering large number of responses is the payment. Most people would not want to answer a long survey for free. Using a well-known crowdsourcing platform also ensures that financial transactions are trustworthy and transparent. We choose CrowdFlower also because it is possible to limit the survey to users that have shown to give quality responses on other studies conducted on CrowdFlower.

For the CrowdFlower platform, each worker has a discrete trust level associated with his or her account ranking from level 1 to level 3. Participants can raise their trust level by completing jobs on the CrowdFlower platform successfully and without failing. A participant fails a study if they ever drop below a certain level of accuracy (70% by default) on pre-defined test questions. However, the exact algorithms for determining user trust levels are internal to CrowdFlower and not visible to us. We are only able to specify the trust level to which our survey is available. Every job on CrowdFlower will have a pre-survey quiz to ensure participants understand the task. Only participants that achieve at least 70% will be allowed to participate in the survey. Furthermore, throughout the survey, there will be a random test question on every survey page to ensure the participant stays focused throughout. Participants gain trust levels by passing these quizzes and test questions and consequently lose trust levels by failing. If they feel a particular test or survey question was unfair, they are able to challenge it and provide a written response that we can monitor in real time and change the questions accordingly while the survey is still live for other participants. We have chosen to limit our survey to only the highest trust level participants at level 3. This is to ensure as much as possible that our responses come from participants who are experienced with the platform and have shown themselves to be trustworthy (ie. do not click through surveys randomly) from past studies on CrowdFlower.

5 METHODS

5.1 Participants

The participants of the survey were users of the CrowdFlower platform from all over the world, including but not limited to countries such as USA, Brazil, Ukraine, Poland, Turkey, Russia, France, Egypt, Mexico, and India. However, as many users choose not to disclose their nationality, we do not have the complete statistics on the locations of the participants. All participants are completely anonymous and only trackable in our experiment via worker ID. The participants are paid $0.02 CDN per comparison once they pass the pre-survey quiz. If a participant drops below 70% accuracy on test questions, they will be kicked from the survey, but still be paid for the comparisons they have done so far. Their responses up to that point will be used. Lastly, participants are free to quit the survey at any time. However, they will only be paid for each page of comparisons for which they have clicked "Submit". After the survey, participants have the option of providing feedback in the clarity of the instructions, easiness of the task, compensation, and overall experience.

There was a total of 1263 trusted annotators from 65 countries, with the majority of the workers coming from Venezuela (24.5%), Brazil (6.7%), Serbia (6.6%), Turkey (6.0%), Russia (5.6%), and Bosnia (5.1%). The trusted annotators had an accuracy of 93.6% on the quiz and the test questions throughout the survey. There were a total of 103 untrusted annotators who were kicked from the survey and 109 people who failed the pre-survey quiz, and thus did not participate in the study. There were a total of 19848 submitted comparisons by trusted annotators. Annotators spent an average of 15 seconds per comparison. This is a reasonable amount of time as the animation clips are being played side by side simultaneously and are 10 to 25 seconds long.

5.2 Procedure

Similar to Baveye et al. [5], we are using a quick-sort algorithm to rank our motion capture animation clips. The reason for this is to cut down the number of comparisons needed to significantly reduce the cost. We have 9 takes for each movement for a total of 181 motion capture clips. Surveying participants on every possible pairwise combination would result in $N(N-1)/2$ comparisons. The idea behind the quick-sort algorithm is that at the first iteration, a specific MoCap is chosen as the pivot. Every other MoCap is compared to the pivot element, creating two subgroups. The assumption is that everything in the subgroup that was considered to have a lower affect than the pivot is also lower in affect than everything in the subgroup higher in affect than the pivot. However, we never compared MoCap from the two subgroups directly. Everything was only compared with the pivot. Then two pivots are chosen within the subgroup in the next iteration. Everything in the first subgroup is then compared with the first subgroup pivot, and vice-versa for the second pivot. This results in an average of $O(nlogn)$ comparisons rather than the full $N(N-1)/2$ for every possible pair-wise comparison. We continue dividing until the size of every subgroup is no more than 5-10. We treat this last subgroup size as having equal affect. We stopped at this subgroup size because we have many improvisation movements that appear to be difficult for people to distinguish a relative rank. Many comparisons at this stage were decided with 3:2 votes, no better than random. We ended up using 9 pivots for valence and 8 pivots for arousal. Using this method, we paid $346.55, saving about $2800-$3000 for this experiment.

The survey begins with an explanation of the concept of valence and arousal with accompanying examples of emotions on both dimensions. The scale and accompanying example for valence is shown in Figure 3. Likewise, the example for arousal is shown in Figure 4. The overlapping terms are to provide additional examples

[4] https://success.crowdflower.com/hc/en-us

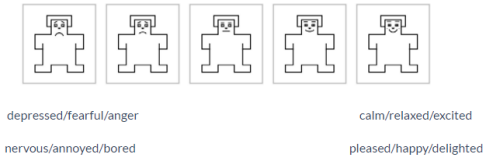


Figure 3: Valence scale example

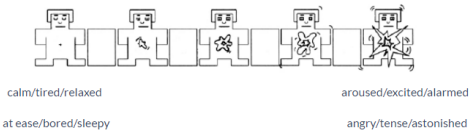


Figure 4: Arousal scale example

of affects that are considered to be low or high valence and arousal. Due to space constraints they were placed on different lines as to maintain a clear distinction of the low VA examples and the high VA examples. The characters in the images are from the SAM (Self-Assessment Manikin) scale [6], a popular set of standard images to convey the spectrum of affect used in research. Even though this is a ranking experiment and not a rating experiment, these images were chosen in addition to the adjectives shown to help the particpant understand the concept of affect.

The participant then takes the pre-survey quiz to ensure they understand the concepts. Animations chosen for the quiz are obvious examples of difference in valence or arousal in the sense that they were extreme comparisons such as high valence high arousal compared with low valence low arousal. An explanation is provided for each question in case the participant picks the wrong answer. The animation itself shows both motion captures simultaneously. The particpant is able to zoom in or out and pan around the scene as they wish to explore the 3D nature of the movement. An example of the interface is shown in Figure 5.

The animation clips used range in length from about 10 to 25 seconds of three different professional actors as described in Section 3.1. Each animation shows a mover performing one of our recorded movements mentioned in Section 3.1. There are 181 different motion capture clips. We collect a total of 5 responses for each comparison. Similar to Baveye et al. [5], we choose an odd number of comparisons so that each pair is guaranteed to have a distinction as to which has the higher affect. However, we choose 5 comparisons as opposed to the 3 comparisons from Baveye to reduce the likelihood of pairs getting close votes by chance or pairs getting ranked in the wrong order. A pair can be ranked incorrectly if two people happened to not pay attention on a particular question. With 5 comparisons, three people need to be not paying attention for a pair to be ranked incorrectly. All of the above procedure is performed twice, once for valence and once for arousal.

Table 1: Distribution of ranks for the 181 motion capture sequences

	Arousal						
180	0	0	1	3	6	21	Valence
150	0	1	1	10	9	9	
120	0	4	6	11	8	1	
90	0	5	16	4	5	0	
60	4	16	6	2	2	0	
30	26	4	0	0	0	0	
Ranks	30	60	90	120	150	180	

6 EXPERIMENT AND RESULTS

6.1 Inter-annotator Reliability

For the ranking experiment, we first look at the inter-annotator reliability. Unreliable responses are filtered by the trust level settings, pre-survey quiz, and random test questions. In the experiment of Baveye et al. [5], they collected 3 responses for each comparison and they measured Inter-annotator reliability using percent agreement, Krippendorff's alpha, Fleiss' kappa, and Randolph's kappa. We conduct the same metrics except Fleiss' kappa in our experiment as a comparison. The results are presented in Table 2. Krippendorff's alpha is a flexible metric for measuring inter-annotator reliability in that it allows for comparions being made by any number of participants and missing data. Randolph's kappa is an alternative to Fleiss' kappa, allowing for more flexibility in the distribution of responses [35]. We did not include Fleiss' kappa because it assumes there will be a certain number of responses for each category. Thus we felt Fless' kappa was not suitable for our experiment.

Similar to Baveye et al., our reliability results indicate that agreement is better than what would have been expected by chance. The agreement from the experiment of Baveye et al. were similar to Malandrakis et al. [27] and Mohammad et al. [30]. Our percent agreement Krippendorff's alpha, and Randolph's kappa were found to be lower than the experiment by Baveye et al., about 5% percent agreement and 0.05 in both Krippendorf's alpha and Randolph's kappa. This suggests it is harder to rank the valence and arousal of motion capture sequences than videos, which is not surprising. Videos contain facial expressions and sound. In the case of the LIRIS-ACCEDE database presented by Baveye et al., the videos are excerpts extracted from movies. Agreement in participant responses may be in part due to recognition of those movies, in which case the participants have the context of the entire movie from which to draw their sense of perceived affect. For example, if all participants recognize the video as an excerpt from a happy movie, they would perhaps be more likely to rank it as higher valence than an excerpt from a sad movie, even if the excerpts might be similar in affect.

In Table 1 we present a distribution of the rankings. As an example, the cell in the bottom left illustrates the number of sequences with a rank of between 0 and 30. The interval size of 30 was chosen to evenly divide all the motion capture clips into sufficient bucket sizes in order to see an overall pattern or trend in the distribution. The sparseness of the top left and bottom right of the graph indicates that there were little to no motion captures that were ranked as either high valence and low arousal or low valence and high arousal. The rankings are concentrated along the $y = x$ line, indicating that valence rises with arousal. This suggests that

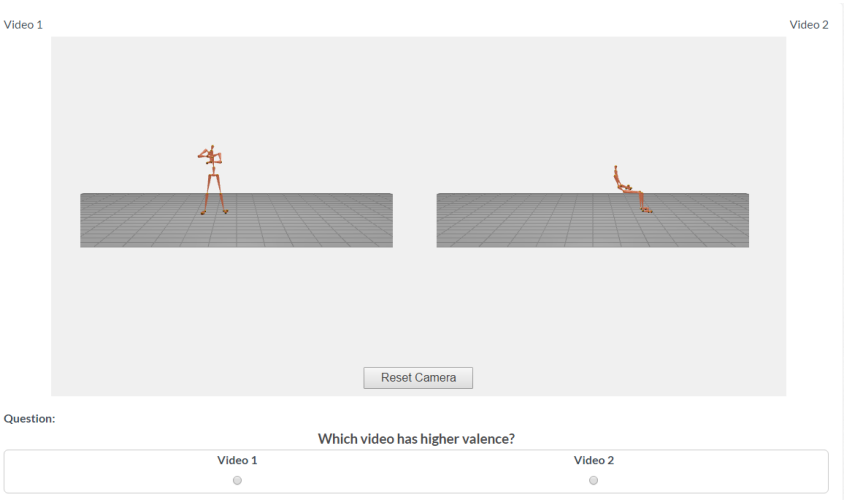


Figure 5: Screenshot of video in survey

	Valence	Arousal
Percent agreement	80.9%	81.2%
Krippendorff's alpha	0.096	0.157
Randolph's kappa	0.370	0.419

Table 2: Inter-annotator reliability

either it is difficult for the actors to portray valence and arousal independently through motion capture, or people in general have difficulty perceiving valence and arousal independently.

6.2 Learning to Rank Method

We use RankNet [7] to train a model to rank motion capture segments based on their valence and arousal levels, that match the rankings provided by the survey. RankNet is a pairwise rank-learning model that consists of a neural-network and a probabilistic loss-function that aims at minimizing the number of rankings in the wrong order. It can be trained using stochastic gradient descent.

6.2.1 RankNet. Burges et al. [7] proposed a probabilistic cost for training systems to learn ranking functions using pairs of training examples. From the training examples they attempt to learn a ranking function that does not map to a particular rank value. For example, if A is ranked higher than B, the system just needs to be able to determine that A is ranked higher, but it does not assign a value to the rank of A or B when learning the training samples. Burges et al. then chose to implement their probabilistic cost function in a neural network. The cost function is a function of the difference of the ranking outputs of two consecutive training samples, ie. what the system thinks is the difference in ranks between two samples versus the actual difference in ranks. In the case of consecutive samples, the true difference in ranks would always be 1. A forward propagation is performed on the first sample, storing the activation of each node in the network and the gradient value. Then the forward propagation is performed on the second sample, again storing the activation and the gradient. The cost then is the difference between the gradients of the two samples. Through learning to minimize the difference of the gradient, RankNet models the training samples in a monotonically increasing order.

6.2.2 Performance Analysis. We evaluate the ranking performance of RankNet using the Goodman-Kruskal gamma [17]. The Goodman-Kruskal gamma is a measure of rank correlation between two variables, the ground truth and the predicted ranks in our case, and is identified by G as follows:

$$G = \frac{(N_s - N_d)}{N_s + N_d} \tag{1}$$

where N_s represents the number of cases that are ranked in the same order on both variables, and N_d represents the number of cases that are ranked in the reverse order. This measure ignores ties. In our case, we did not have any ties. A G close to 1 shows strong agreement in rankings between the two variables, while a G close to -1 shows a strong disagreement in the rankings. A G close to 0 shows the rankings are independent and not associated with each other.

We use a neural network with two hidden layers, with 600 and 300 Rectifier Linear Units (ReLU) each, respectively and chosen from cross-validation to give the best results. We train the model on different subsets of the data based on the movement type and the performer. We also experiment with different window sizes. The windows of frames are concatenated in a single feature vector and then fed into RankNet. This is to cover different lengths of temporalities of movement. We stop learning after 3000 epochs. This value was chosen because we notice that after 3000 epoches the

Table 3: The Goodman-Kruskal Gamma for Valence Rankings ($p < 0.0001$)

	Window Sizes (all performers)		
	W = 1	W = 3	W = 12
Walking	0.664982	**0.695**	0.668
Sitting	**0.543481**	0.535	0.537
Hugging	0.205258	0.217	**0.245377**
Pointing	**0.548836**	0.544	0.548
Lie Down	0.600258	**0.602**	0.589
Free Improv	0.603071	**0.608**	0.574696
Improv Facing	**0.731329**	0.714	0.713
Improv Static	0.274803	**0.291**	0.262
All Movements	**0.582195**	0.561	0.536

Table 4: The Goodman-Kruskal Gamma for Arousal Rankings ($p < 0.0001$)

	Window Sizes (all performers)		
	W = 1	W = 3	W = 12
Walking	0.757263	**0.763**	0.741
Sitting	**0.852446**	0.846	0.789
Hugging	**0.938629**	0.914	0.878
Pointing	**0.903560**	0.884	0.853
Lie Down	**0.883156**	0.878	0.821
Free Improv	**0.863757**	0.862	0.826
Improv Facing	**0.878537**	0.859	0.831
Improv Static	**0.866419**	0.834	0.832
All Movements	**0.6271973**	0.623	0.606

loss value was no longer changing. We use 10-fold cross-validation on our training data to report the model accuracy. The G values are summarized in Table 3 For valence rankings, and in Table 4 for arousal rankings. The results from models that are trained on the data from only one performer each time is reported in Table 5.

However, there are a few factors of which we should make note. For the walking movements, we can consider a single walking cycle to be regarded as a single instance of that specific movement type and we have a few cycles of those per affect and mover. For all other movement types we only have a single instance. So however many of the frames for validation we set aside, those frames belong to the same performance. Thus, any conclusions about the generalization of the model beyond this dataset is unreliable. In other words, with the data we have, we are unable to test our machine learning models properly, except maybe for the walking movements.

The fact that many of the highest gammas came from a window size of 1 suggests that this method, or at the very least this configuration of neural network, is more effective for postures than movement. The high gamma for performer 3 suggests that she was able to act out her intended affect most effectively. However, this may also be due to performer 3 only acting in the walking movements, which as was just mentioned is the most reliable movement. Furthermore, we see that arousal is much more consistent in having a higher gamma in the window size of 1 as well as having higher gamma than valence across the board. This is again in accordance with the suggestion that arousal is easier to recognize and learn, both by human viewers as well as a machine. Looking at the different movements in Table 3, we see that that the hugging and improv static movements have resulted in a much lower gamma than the others. Compared to the others, these two types have the least amount of movement, suggesting that changes in postures or some other movement characteristics that we are unaware of are needed in order to recognize valence with a machine.

7 CONCLUSION

We first conducted a groundtruthing experiment on CrowdFlower, a crowdsourcing platform, where we surveyed participants from all over the world. We then constructed a machine learning model using RankNet to predict the relative ranks for pairs of motion capture clips in a variety of movements. These results and contributions are a first step in future experiments in affect estimation with large amounts of motion capture data, leading to use cases such as database labelling and movement generation.

Table 5: The Goodman-Kruskal Gamma for Valence and Arousal Rankings - Individual Performers ($p < 0.0001$, $W = 3$)

	P1	P2	P3
Valence	0.496008	0.587459	**0.647168**
Arousal	0.638740	0.667621	**0.750769**

In the CrowdFlower survey, participants were given the definition of valence and arousal as well as training examples. They then watched pairs of motion capture clips and were required to answer which within the pair had higher valence or arousal. These rankings were used in a quicksort algorithm to establish the relative ranks of 181 motion capture clips, containing 9 different movements. The survey showed better than random percent agreement but slightly lower compared to video-based affect recognition surveys. This is to be expected as it is not surprising that the lack of facial expression would result in an increased difficulty to distinguish affect.

Our experiments with RankNet show that it is possible to build machine learning models to learn the relative ranking of motion capture clips, with Goodman-Kruskal gamma of 0.62 to 0.93 in case of arousal rankings, and 0.24 to 0.73 in case of the valence rankings. The performance highly depends on the movement type, as well as the performers in that, having consistent movement patterns in the training data (i.e., same movement type and same performer) improve the chances that RankNet can effectively learn from them.

Another observation is that ranking predictions for valence were more or less lower than those for arousal. This is consistent with the findings from the survey in which higher number of people failed the valence pre-survey quiz, as well as the lower post-survey ratings for valence.

We understand the current limitations of our learning to rank approach. First, there is only one repetition of each combination of movement type/performer/affective state. This limits the amount of variation in the data and the ability of the model to generalize. We plan to gather more data and perform larger ground-truthing studies. Second, RankNet does not take into account the temporality of the data. As we see in the results, the model achieve higher precision, in most cases, with a window size of 1, which means that the model is essentially learning the postures alone and not the dynamic qualities of the movement.

We are interested in exploring training more effective machine learning models such as Recurrent Neural Networks. Furthermore, as mentioned in Section 6.2.2, the hugging and improv static movements have resulted in significantly lower gamma compared to the other movements. An area of exploration is establishing a set of rules or characteristics for different movement types that will determine whether the movement would be easy for affect estimation by machines. It may also be worth considering time-series analysis techniques instead of neural networks to try to account for the temporal aspect of movement.

ACKNOWLEDGMENTS

We would like to thank the Social Sciences and Humanities Research Council for funding in this experiment.

REFERENCES

[1] Omid Alemi, William Li, and Philippe Pasquier. 2015. Affect-expressive movement generation with factored conditional Restricted Boltzmann Machines. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*. IEEE, 442–448.

[2] Sharifa Alghowinem, Roland Goecke, Michael Wagner, Gordon Parkerx, and Michael Breakspear. 2013. Head pose and movement analysis as an indicator of depression. In *Proceedings of the 2013 Conference on Affective Computing and Intelligent Interaction*. IEEE Computer Society, 283–288.

[3] Michael Argyle. 1988. *Bodily Communications*. Methuen & Co. Ltd.

[4] Yoann Baveye, Emmanuel Dellandrea, Christel Chamaret, and Liming Chen. 2014. From crowdsourced rankings to affective ratings. In *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*. IEEE, 1–6.

[5] Yoann Baveye, Emmanuel Dellandrea, Christel Chamaret, and Liming Chen. 2015. Liris-accede: A video database for affective content analysis. *IEEE Transactions on Affective Computing* 6, 1 (2015), 43–55.

[6] Margaret M Bradley and Peter J Lang. 1994. Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of behavior therapy and experimental psychiatry* 25, 1 (1994), 49–59.

[7] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 89–96.

[8] Antonio Camurri, Ingrid LagerlÃűf, and Gualtiero Volpe. 2003. Recognizing emotion from dance movement: comparison of spectator recognition and automated techniques. *International Journal of Human-Computer Studies* 59, 1-2 (2003), 213–225. https://doi.org/10.1016/S1071-5819(03)00050-8

[9] Ginevra Castellano, Santiago D Villalba, and Antonio Camurri. 2007. Recognising Human Emotions from Body Movement and Gesture Dynamics. In *Affective Computing and Intelligent Interaction*. Lecture Notes in Computer Science, Vol. 4738. Springer Berlin Heidelberg, 71–82. http://dx.doi.org/10.1007/978-3-540-74889-2_7

[10] Beatrice de Gelder. 2006. Towards the neurobiology of emotional body language. *Nature Reviews Neuroscience* 7, 3 (2006), 242–249. https://doi.org/10.1038/nrn1872

[11] Marco de Meijer. 1989. The contribution of general features of body movement to the attribution of emotions. *Journal of Nonverbal Behavior* 13, 4 (1989), 247–268. https://doi.org/10.1007/BF00990296

[12] Jianyu Fan, Kıvanç Tatar, Miles Thorogood, and Philippe Pasquier. 2017. Ranking-based emotion recogntion for experimental music. *Intern* (2017).

[13] Beat Fasel and Juergen Luettin. 2003. Automatic facial expression analysis: a survey. *Pattern recognition* 36, 1 (2003), 259–275.

[14] Johnny RJ Fontaine, Klaus R Scherer, Etienne B Roesch, and Phoebe C Ellsworth. 2007. The world of emotions is not two-dimensional. *Psychological science* 18, 12 (2007), 1050–1057.

[15] Nickolaos Fragopanagos and John G Taylor. 2005. Emotion recognition in human-computer interaction. *Neural Networks* 18, 4 (2005), 389–405.

[16] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research* 4, Nov (2003), 933–969.

[17] Leo A Goodman and William H Kruskal. 1954. Measures of association for cross classifications. *J. Amer. Statist. Assoc.* 49, 268 (1954), 732–764.

[18] Sebastian Grassia. 1998. Practical parameterization of rotations using the exponential map. *Journal of graphics tools* 3, 3 (1998), 29–48.

[19] Shan He, Shangfei Wang, Wuwei Lan, Huan Fu, and Qiang Ji. 2013. Facial Expression Recognition Using Deep Boltzmann Machine from Thermal Infrared Images. In *Proceedings of the 2013 Conference on Affective Computing and Intelligent Interaction*. IEEE Computer Society, 239–244.

[20] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. (2000).

[21] Martin Inderbitzin, Aleksander Väljamäe, Jose Maria Blanco Calvo, Paul FMJ Verschure, and Ulysses Bernardet. 2011. Expression of emotional states during locomotion based on canonical parameters. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. IEEE, 809–814.

[22] Shihoko Kamisato, Satoru Odo, Yoshino Ishikawa, and Kiyoshi Hoshino. 2004. Extraction of Motion Characteristics Corresponding to Sensitivity Information Using Dance Movement. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 8, 2 (2004), 168–180. http://dblp.uni-trier.de/db/journals/jaciii/jaciii8.html#KamisatoOIH04

[23] Asha Kapur, Ajay Kapur, Naznin Virji-Babul, George Tzanetakis, and Peter Driessen. 2005. Gesture-Based Affective Computing on Motion Capture Data. In *Affective Computing and Intelligent Interaction*. Lecture Notes in Computer Science, Vol. 3784. Springer Berlin Heidelberg, 1–7. https://doi.org/10.1007/11573548_1

[24] Andrea Kleinsmith and Nadia Bianchi-Berthouze. 2013. Affective Body Expression Perception and Recognition: A Survey. *IEEETransactions on Affective Computing* 4, 1 (2013), 15–33. https://doi.org/10.1109/T-AFFC.2012.16

[25] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[26] Yingliang Ma, Helena M Paterson, and Frank E Pollick. 2006. A motion capture library for the study of identity, gender, and emotion perception from biological motion. *Behavior research methods* 38, 1 (2006), 134–141.

[27] Nikolaos Malandrakis, Alexandros Potamianos, Georgios Evangelopoulos, and Nancy Zlatintsi. 2011. A supervised approach to movie emotion tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2376–2379.

[28] Albert Mehrabian. 1996. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology* 14, 4 (1996), 261–292.

[29] Albert Mehrabian and James A Russell. 1974. *An approach to environmental psychology*. M.I.T. Press.

[30] Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence* 29, 3 (2013), 436–465.

[31] Mihalis A Nicolaou, Hatice Gunes, and Maja Pantic. 2011. Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space. *Affective Computing, IEEE Transactions on* 2, 2 (2011), 92–105. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5740839

[32] Michael Nixon, Ulysses Bernardet, Sarah Alaoui, Omid Alemi, Ankit Gupta, Thecla Schiphorst, Steve DiPaola, and Philippe Pasquier. 2015. MoDa: an open source movement database. In *Proceedings of the 2nd International Workshop on Movement and Computing*. ACM.

[33] Hanhoon Park, Jong-Il Park, Un-Mi Kim, and Woontack Woo. 2004. Emotion Recognition from Dance Image Sequences Using Contour Approximation. In *Structural, Syntactic, and Statistical Pattern Recognition*. Lecture Notes in Computer Science, Vol. 3138. Springer Berlin Heidelberg, 547–555.

[34] Frank E Pollick, Vaia Lestou, Jungwon Ryu, and Sung-Bae Cho. 2002. Estimating the efficiency of recognizing gender and affect from biological motion. *Vision Research* 42, 20 (2002), 2345 – 2355. https://doi.org/10.1016/S0042-6989(02)00196-7

[35] Justus J Randolph. 2005. Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa. *Online submission* (2005).

[36] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. 2011. SortNet: Learning to rank by a neural preference function. *IEEE transactions on neural networks* 22, 9 (2011), 1368–1380.

[37] James A Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology* 39, 6 (1980), 1161–1178.

[38] Ali-Akbar Samadani, Rob Gorbet, and Dana Kulic. 2014. Affective Movement Recognition Based on Generative and Discriminative Stochastic Dynamic Models. *Human-Machine Systems, IEEE Transactions on* 44, 4 (2014), 454–467.

[39] Daniel L Schacter, Daniel T Gilbert, and Daniel M Wegner. 2009. *Introducing psychology*. Macmillan.

[40] Jan Van den Stock, Ruthger Righart, and Beatrice De Gelder. 2007. Body expressions influence recognition of emotions in the face and voice. *Emotion* 7, 3 (2007), 487–494.

[41] Harald G Wallbott. 1998. Bodily expression of emotion. *European journal of social psychology* 28, 6 (1998), 879–896.

[42] Georgios N Yannakakis and Héctor Perez Martínez. 2015. Ratings are Overrated! *Frontiers in ICT* 2, 13 (2015). https://doi.org/10.3389/fict.2015.00013

Appendix E

m+m: A novel Middleware for Distributed, Movement-based Interactive Multimedia Systems

This appendix presents content from the following paper:

Ulysses Bernardet, Dhruv Adhia, Norman Jaffe, Johnty Wang, Michael Nixon, Omid Alemi, Jordon Phillips, Steve DiPaola, Philippe Pasquier, and Thecla Schiphorst. 2016. M+m: A Novel Middleware for Distributed, Movement Based Interactive Multimedia Systems. In Proceedings of the 3rd International Symposium on Movement and Computing (MOCO '16). ACM, Article 21, 9 pages. DOI: https://doi.org/10.1145/2948910.2948942

m+m: A novel Middleware for Distributed, Movement based Interactive Multimedia Systems

Ulysses Bernardet
Simon Fraser University
Vancouver, Canada
ubernard@sfu.ca

Dhruv Adhia
H Plus Technologies
Vancouver, Canada
dhruv@hplustech.com

Norman Jaffe
Vecima Networks
Vancouver, Canada
turing@shaw.ca

Johnty Wang
McGill University
Montreal, Canada
johnty.wang@mail.mcgill.ca

Michael Nixon
Simon Fraser University
Vancouver, Canada
mna32@sfu.ca,

Omid Alemi
Simon Fraser University
Vancouver, Canada
oalemi@sfu.ca

Jordon Phillips
Simon Fraser University
Vancouver, Canada
jjp14@sfu.ca

Steve DiPaola
Simon Fraser University
Vancouver, Canada
sdipaola@sfu.ca

Philippe Pasquier
Simon Fraser University
Vancouver, Canada
pasquier@sfu.ca

Thecla Schiphorst
Simon Fraser University
Vancouver, Canada
thecla@sfu.ca

ABSTRACT
Embodied interaction has the potential to provide users with uniquely engaging and meaningful experiences. m+m: Movement + Meaning middleware is an open source software framework that enables users to construct real-time, interactive systems that are based on movement data. The acquisition, processing, and rendering of movement data can be local or distributed, real-time or off-line. Key features of the m+m middleware are a small footprint in terms of computational resources, portability between different platforms, and high performance in terms of reduced latency and increased bandwidth. Examples of systems that can be built with m+m as the internal communication middleware include those for the semantic interpretation of human movement data, machine-learning models for movement recognition, and the mapping of movement data as a controller for online navigation, collaboration, and distributed performance.

Author Keywords
Real-time interaction; middleware; movement;

ACM Classification Keywords
C.3. SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS]: Real-time and embedded system, D.2.11. Software Architectures: Domain-specific architectures

MOCO'16, July 05-06, 2016, Thessaloniki, GA, Greece
 ISBN 978-1-4503-4307-7/16/07..$15.00
DOI: http://dx.doi.org/10.1145/2948910.2948942

INTRODUCTION
We can observe converging trends in human-computer interaction, cognitive science, and the consumer market: firstly, *affective computing*, the research and development of software systems that can recognize, interpret, process, and ultimately harness affective responses [16], has become a mainstream topic. Secondly, cognitive science has shown an increasing interest in *embodied cognition*, i.e. the proposition that the mind "is not only connected to the body but that the body influences the mind" [26]. Thirdly, in the consumer market we can observe a trend towards the engagement of individual non-experts in the self-monitoring and -analysis of biological, physical, behavioral, or environmental information referred to as "*quantified self*" [24]. What these trends share is the notion that to better understand humans, and/or to build better technology, we need to take into account the body, and with it, movement.

This motivation is met at the technological level by recent developments in the hardware and software domains. In the former we observe a proliferation and democratization of real-world behavior and movement sensors – on the one hand in the form of affordable sensors such as Microsoft Kinect, Structure Sensor, Wii Balance Board and Remote, and Leap Motion, and on the other hand through wearable technology [25]. In the latter, the software domain, systems for making inferences based on movement such as gesture recognizer

[6], or through the application of the Laban Movement Analysis [11], a well-established system for describing movement, have gained traction.

If our goal is to build real-time, distributed interactive systems that deploy heterogeneous sensors and effectors, we need the means for recording and sensing, storing and retrieving, analyzing and understanding, and displaying, sonifying, and visualizing movement data. And, crucially, we need a way to connect these elements together, and have them communicate with each other. In this manuscript we describe the "m+m: Movement + Meaning" software framework that, broadly speaking, enables users of different domains and levels of expertise to construct real-time, interactive systems that are based on movement. m+m caters to users from a range of backgrounds including, but not limited to, performance art, computer engineering, science, and health technology. At the software engineering level m+m is based on the well-established Yet Another Robot Platform (YARP) [15]). The acquisition, processing, and rendering of movement data can be local or distributed, real-time or off-line, and m+m provides a range of ready-made interfaces to devices, and existing software frameworks. A graphical user interface (GUI) provides a tool for managing and monitoring nodes in the network.

RELATED WORK

The requirements flagged above are met to varying degrees by existing software solutions. Here we will give a brief overview of existing middleware services, frameworks, communication libraries, and integrated packages. A *middleware* service is as a general-purpose service that sits between platforms and applications, and that is defined by the Application Programming Interfaces (APIs) and protocols it supports [5]. A number of classification schemas for middleware exist, e.g. [17] distinguishes between Transactional middleware for distributed synchronous transactions, Procedural middleware to execute Remote Procedure Calls (RPC), Message-oriented middleware that provide communication through messages (e.g. IBM WebSphere MQ[1], Apache ActiveMQ[2]) and object-oriented middleware that extends RPC with concepts from object-orientation (e.g. Java Remote Method Invocation (RMI[3]), and Common Object Request Broker Architecture (CORBA[4]). The advantages of most of these middleware frameworks are that they are well supported, facilitate development, and provide a solid basis for setting up and managing communication between nodes. The downside is that many of them are closed source, and have a notoriously large overhead and steep learning curve. *Frameworks* such as Processing [20], openFrameworks[5], MAX[6], and Pure Data [18] are widely used in the artistic and human-computer interaction community. These frameworks put the emphasis on output and rendering, and, while some of them provide built-in networking capabilities, they are generally confined to point-to-point networking and limited in capacity and parallelism. Last but not least, there exists a number of open source and commercial *communication libraries* that differ in the supported protocols, platforms, and level of abstraction at which they are implemented. Examples of open source libraries include Open Sound Control (OSC[7]), Torque Network Library (OpenTNL[8]), POCO C++[9], ADAPTIVE Communication Environment (ACE[10]), and ENet[11], while examples of commercial libraries are RakNet[12], and Zoidcom network[13]. Most of these libraries are agnostic as to what content they transport, in the sense that they do not provide protocol definitions and do not provide built-in means to setup, manage, and monitor connections between nodes. *Integrated solutions* closest to the approach presented in this manuscript include the StreamInput advanced sensor processing and user interaction application programming interface (API) developed by the Khronos working group[14].

In the domains of pervasive and ubiquitous computing a number of comprehensive middleware systems have been developed. Some of these systems are specialized e.g. for ubiquitous tracking, where data from spatially distributed and heterogeneous tracking sensors need to be integrated, such as the CORBA based Ubitrack framework [19] and its predecessor DWARF [13]. Other systems have wider application domains such the Proximity Toolkit that supports proxemics based interactions [14], frameworks for building distributed tangible and multi-modal interfaces such as Ensemble [7] and DynaMo [2], respectively, and the Stanford Interactive Room Operating System (iROS), a general purpose software framework which allows applications to communicate with each other and with user interface devices in a dynamically configurable way [9]. Possibly closest to m+m in terms of scope and design philosophy is the real-time Java-based middleware OSA+ [22], supports the construction of distributed, heterogeneous, and highly scalable systems.

The development of m+m is motivated by the set of specific requirements for the development of the types of systems outlined initially. The middleware should be a largely self-sufficient system, enabling users with little technical background to build interactive systems. Hence, m+m needs

[1] http://ibm.com/software/products/en/ibm-mq
[2] http://activemq.apache.org
[3] http://goo.gl/SqNX33
[4] http://omg.org/spec/CORBA/
[5] http://openframeworks.cc
[6] http://cycling74.com/products/max/
[7] http://opensoundcontrol.org
[8] http://opentnl.org
[9] http://pocoproject.org
[10] http://cs.wustl.edu/~schmidt/ACE.html
[11] http://enet.bespin.org
[12] http://jenkinssoftware.com
[13] http://zoidcom.com
[14] http://khronos.org/streaminput/

to be able to provide turnkey solutions, i.e. not merely an API. Hand in hand with this requirement goes the need to provide a library of interfaces to established, predominantly movement data acquisition sensors, ranging from Kinect to professional motion capture systems. To facilitate interoperability, a standardized protocol, specifically tailored to movement-based data has to be an integral part of the middleware. This is a key feature, that is – basic network libraries are missing. The middleware needs to provide high bandwidth data transmission that allows data to be streamed raw, or minimally processes sensor information via processing components in real time. In a fluid, exploration- and development-oriented deployment scenario, decoupling of components is essential, allowing users to connect and disconnect nodes at run time. Last but not least, the development of the middleware was motivated by the desire to provide users with an easy to install, and open source system.

M+M ARCHITECTURE

Conceptual framework

m+m is endorses a component-based architecture of logically independent entities, and is based on the well-established open source middleware "YARP" [15]. Key features of the m+m middleware are portability between different platforms, a small footprint in terms of computational resources, and high performance regarding latency and bandwidth. The first two properties are achieved by m+m being cross-platform, with support for all major operating systems (Windows, MacOS, and Linux), the core binary distribution being portable (for convenience, binary installers are provided), and m+m having a small footprint (the windows distribution requires less than 100.0MB disk space). The high performance in terms of latency and bandwidth is achieved by two main mechanisms: the middleware itself does not handle any communication, but rather establishes direct point-to-point communication between end-nodes. Secondly, all communication is based directly on native protocols with as little overhead as is possible. Depending on the specific needs and system topology, communication can be done via TCP/IP, UDP, or shared memory. The properties listed above are key advantages over other middleware platforms such as Apache ActiveMQ[15], or ROS[16]. Additionally, m+m, by virtual of being based on YARP, provides bindings for multiple languages (C++, Perl, Python, Java), and comes with a number of basic interfaces to hardware devices such as microphones and cameras.

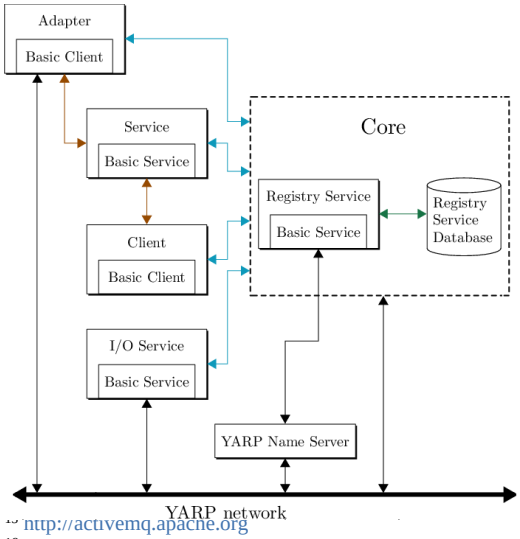


Figure 1: Logical organization of an m+m system. Installation. Brown lines represent client-service communication, the blue lines represent communication with the Registry Service and the black lines represent YARP communication paths.

m+m middleware

All m+m programs utilize YARP to facilitate communication – it provides one-to-many output and many-to-one input mechanisms, as well as a network-based name server (Figure 1). These input and output mechanisms are implemented via "mini-server" code that is a fundamental component of YARP. The "YARP network" represents the aggregated TCP/IP, UDP and shared memory connections that exist when YARP is active – YARP itself does not use any special protocols and can operate over a variety of physical networks. What m+m provides is a standardized client-service mechanism, a set of naming conventions for YARP ports and a centralized database that is used to locate services within the YARP network. The YARP Name Server is used to obtain the physical network address of each m+m channel, given the name of the channel. Once the network address is known, all communication between entities in m+m is via either TCP/IP or UDP packets, using YARP low-level mechanisms to manage the connections. Services perform a sequence of requests and responses with the Registry Service when they start, in order to be accessible from other m+m entities. Once started, they can receive requests from client applications, data streamed via their input channels, external sensors or generated algorithmically, and transmit data via their output channels, external transducers or files. Additionally, they will receive periodic requests from the Registry Service, inquiring as to their "health" and availability.

m+m Components

Sensors The term "Sensor" refers to a wide range of components providing input to the middleware. Technically a "sensor" ranges from a hardware device (e.g. camera) to high-level processing entities that extract semantically meaningful information from a physical sensing device. Currently, the following sensors are supported: all native YARP devices (serial, video, audio, etc.), Microsoft Kinect (version 1 and 2), Leap Motion[17], AnTS Overheard tracking [3], several motion capture systems (organic motion

[15] http://activemq.apache.org

[16] http://ros.org

[17] http://leapmotion.com

OpenStage, OptiTrack NatNet, Vicon DataStream), biosignals acquisition hardware (BITalino, Thought Technology ProComp2), and sensor data from iOS devices.

Effectors An effector is a component that produces output perceivable by users. As with the sensors, the effectors are interfaced at different semantic levels and are equipped with different levels of autonomy. Currently the following effectors are supported: iDanceForms[18], game engines Unity 3D (unity3d.com) and Unreal Engine[19], and SmartBody (via an ActiveMQ adaptor).

Processing Components The role of a processing component is to mediate between inputs into the system and output generated by the system. Examples of functionality implemented in processing components include feedforward and feedback controllers, psychological models, cognitive architectures, artificial neural networks, machine learning modules, and gesture classifiers. Feature extraction modules are a type of processing component that play a central role in the interpretation of meaning from movement information and can be used e.g. for on-line semantic inferences based on the Laban Movement Analysis [12] that has been successfully used to train dancers, animate characters and automatically segment motion capture input. Currently supported processing components include modules implemented directly in C++, perl or python (via SWIG based language bindings to YARP), MathWorks Matlab and Simulink[20], Processing[21], MAX (cycling74.com/products/max/), large-scale neuronal system simulator iqr[22] [4], and openFrameworks.

m+m GUI (Manager Utility)

The strong separation of components into individual executables in m+m can lead to a usability penalty. To mitigate this issue, m+m provides a graphical tool for managing the system components and the connections between them. The m+m Manager Utility application provides a GUI-based view of the state of connections, services and clients within the installation (Figure 2). The m+m manager Utility application displays a single window view of the connections within a YARP network, with features designed to make management of an m+m installation easier. In the diagram of the network topology, standard YARP components, m+m simple clients, m+m services, and m+m adapters are identified by their type (input or output), IP address, and the number and name of their port. Tags e.g. "S" and "C" are used to identify the type of component in the diagram. Connections between ports are shown as lines with one of three thicknesses and one of three colors. From thinnest to thickest lines, the representations indicate: simple YARP network connections; connections between input/output services; and connections between clients and services. Complementary to the thickness of the edges, the colors indicate whether the connection is TCP/IP (teal), UDP (purple), or shared memory (orange). Next to creating and deleting connections, the m+m GUI provides users with numerous ways to manage their m+m system. Using the tool, users can restart and stop running m+m services and adapters, start and restart the Registry services, and launch registered m+m components. Key managerial features are the ability to display information about a service or adapter, enabling and disabling the collection of service

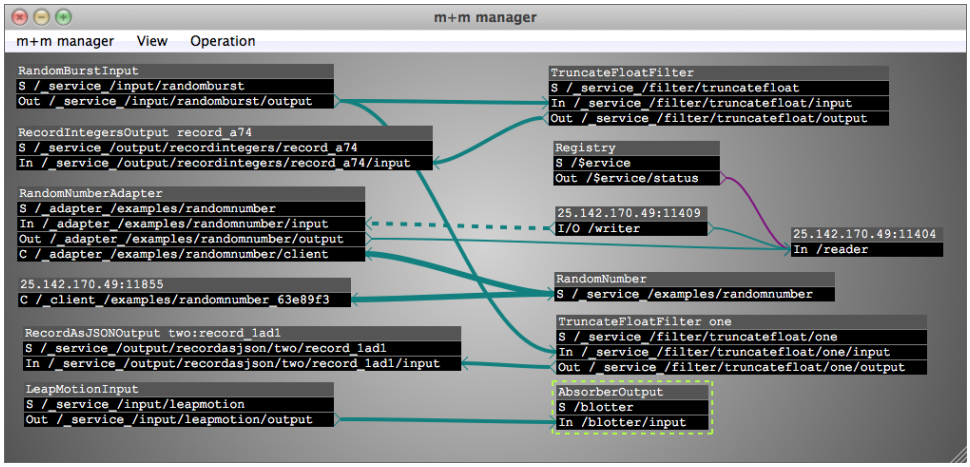


Figure 2: The m+m graphical user interface (GUI) is used to create and manage connections between nodes connected to the m+m middleware, to start and stop m+m services, and to display information about static and dynamic properties of nodes.

[18] http://credo-interactive.com
[19] http://unrealengine.com
[20] http://mathworks.com
[21] http://processing.org
[22] http://iqr.sf.net

metrics about the activity on each port of the service (e.g. the number of bytes and number of messages sent to and from the port).

Registry Service

The Registry Service application is a background service that is used to manage other services and their connections. Its primary purpose is to serve as a repository of information on the active services in an m+m system. The Registry Service provides this feature by maintaining searchable descriptions of the active input/output services, hence allowing application to find and connect to those services. Within the m+m system, the central Registry Service plays a key role in enhancing the manageability of complex distributed systems with potentially large numbers of components. Without such a system, the user has to manually keep track of what system is running where, and what services are provided on which port.

m+m Utilities

The utility programs that are part of m+m provide access to the processes that are running in the m+m installation. Although native YARP commands can be used to manage the network connections, it is recommended that the more specialized m+m tools be used to avoid inconsistencies. These m+m utilities include tools for the inspection of activities and components of the m+m system, and to provide displays for: the active services in the m+m installation; the clients for services that have YARP network connections with persistent state; the primary channels belonging to a service matching a given criterion; information on requests for one or more active services; and measurements for the channels of one or more active services. Additionally, m+m provides applications that allow recording streams of YARP values to an external file. These applications respond to the standard Output service requests and can be also be used as standalone data generators.

Integration of the movement database "MoDa"

The Movement Database (MoDa) is used to store motion capture data associated with video, and qualitative annotations at different semantic levels. Database information can be queried by, and streamed to, any node attached to the m+m middleware. Conversely, nodes in the m+m network can request data to be stored in the database. MoDa is built around a Ruby on Rails application that stores info in a MySQL database (mysql.com). Through the web front-end, researchers can both access and upload movement data. Each file or group of files can also be viewed in the accompanying "MoVa" movement visualizer [1]. MoDa provides programmatic access through the use of a standardized RESTful API that allows communication using HTTP message passing. As the middleware server has all the appropriate API requests programmed into it, a middleware client can make requests to the server in an abstract manner. Once a user authenticates through the client, they are able to communicate with MoDa.

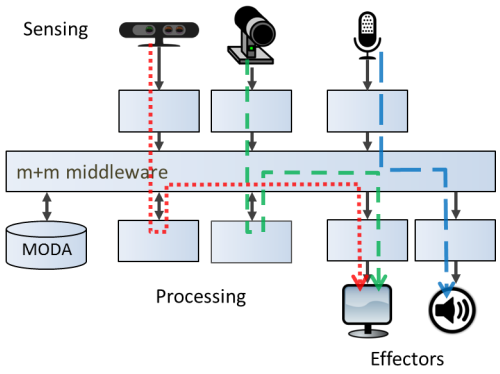


Figure 3: Illustration of the "multipath" capability of an m+m based system.

Standard protocols

Lacking standard protocols for representing messages requires users to define custom data structures. This potentially impairs interoperability and ease of use because the protocols can vary between users and between applications. As mentioned above, the m+m Registry Service allows users to query the syntax and semantics of messages. Complementary to this service, m+m uses a set of standard, interoperable sensor protocols. The basic message packaging in YARP is in the form of "Bottles" that can be containers for primitive types, lists, and "Properties" i.e. associations between tags and values. Bottles are recursive in that they can contain Bottles themselves. Based on the mechanisms provided by YARP, m+m specifies structures of sensor protocols for Kinect, Leap, Vicon, and AnTS tracking. The Extended Backus–Naur Form of these protocols is as follows:

```
x = float; y = float; z = float; w = float; id =
string; tag = string;
quaternion = w, x, y, z;
position3D = x, y, z;
position2D = x, y;
joint = tag, position3D , quaternion;
skeleton = id, joint, {joint};
palm = joint, {joint};
user = id, position2D;
Kinect = skeleton;
Leap = palm;
Vicon = skeleton, {skeleton};
AnTS = user, {user};
```

Prototypical system

One of the main advantages of building systems based on m+m is its "multipath" feature, i.e. the ability to build systems where the information from the same source is concurrently processed by multiple instances without the processing instances interfering with each other or altering the information source (within the constraints of the overall network bandwidth). Figure 3 illustrates such a multipath

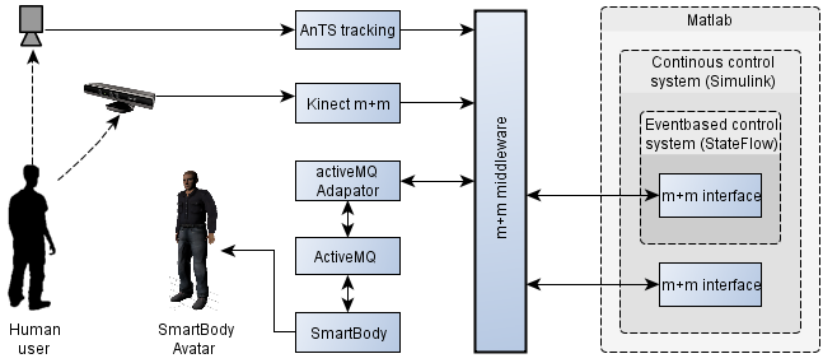


Figure 4: m+m based system for real-time interaction between human and a virtual character

system. Using a Microsoft Kinect, a microphone, and a camera, information from the environment is acquired. The information from the first two sensors is then passed through processing components, e.g. for posture and gesture analysis. Unprocessed, in the case of the microphone, and processed information is then fed to the effectors, such as a monitor and a speaker respectively. It is important to note that all information is acquired, transmitted, processed, and displayed in parallel, without mutual dependencies between the components.

EVALUATION

The subsequent sections give an overview of a number of distributed, real-time interactive systems that have been built using the m+m middleware. Real-time in the current context means that the data is processed and transmitted within the limits of what is perceivable as a delay by an observer. Generally, this ability depends on the processing speed of the nodes (e.g. the motion capture system) combined with the transportation bandwidth and lag. Each of these systems serves to illustrate specific aspects of the m+m middleware.

Distributed real-time mixed-reality dance performance

This example highlights the use of m+m in a performance artistic context where e.g. several dancers co-perform across spatially distributed locations, or a choreographer interacts with performers in real-time over large distances. The concrete system we describe here connects motion acquisition systems at two locations: firstly, in Montreal, at the Computer Research Institute of Montréal (CRIM), hand and finger movement is recorded using a Leap Motion controller. Secondly, motion capture data from a Vicon motion capture system located in Vancouver at the Emily Carr University provides the movements of two dancers: one dancer with a full motion capture suit and a second dancer with wands. Data from the first dancer is mapped onto a humanoid character in the virtual space and the second dancer's movements drive ribbons in the same space. Concurrently information from the Montreal site determines

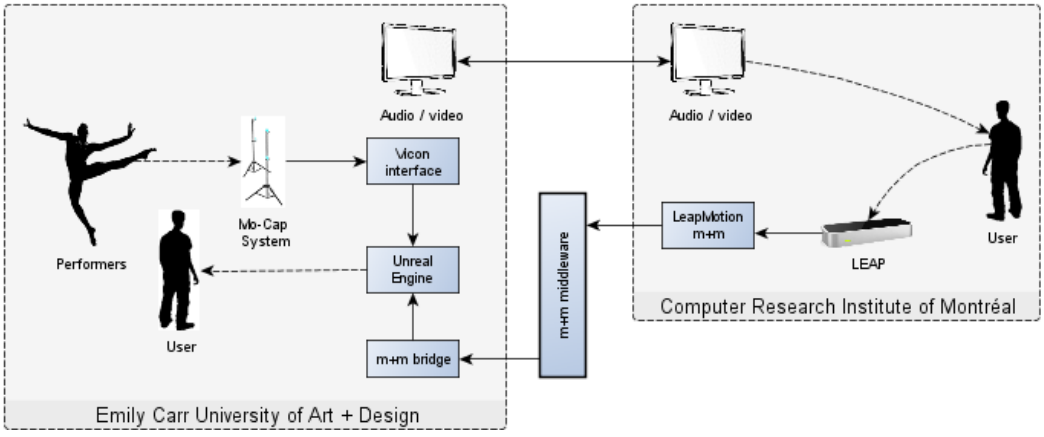


Figure 5: Architecture of the system for a real-time mixed-reality dance performance.

the locations of spotlights in the virtual space. A cluster of computers located in Vancouver provides the logical backbone for the performance: one system acts as the m+m Registry server, one acts as the YARP name registry, another is the motion capture source, one is the m+m system monitor and the last system generates the visual representation of the virtual space. The computer-based communication is via a secure software-based VPN system, that creates a single subnet between the participating computers. The application that generates the virtual space receives its input via a high-speed connection from the local motion capture system and a "bridged" connection from CRIM which is provided by an m+m application that interfaces to the Leap Motion controller. The use of the m+m backbone allows network-address-independent references to the sources and destinations of the messages as well as real-time monitoring and control of the communication paths; the m+m applications dynamically establish their identity and locations, register themselves with the globally-visible m+m Registry server and then are connected at the time of the performance via the m+m system monitor. By using m+m the participants in the performance are able to quickly setup and execute the performance.

System for real-time, real-world interaction between humans and virtual characters

This example illustrates the use of m+m in the construction of a distributed system in which a human is interacting with a virtual character – a realistic 3D representation of a human – in real-time. Such systems can be used e.g. in education and training in performing arts, psychological training and counseling, sports training etc. [8]. In the concrete case elaborated here, the system is used to develop a biologically and psychologically grounded cognitive architecture for the control of nonverbal behavior of a virtual humanoid character during dynamic interactions with human users [21]. Figure 4 illustrates how the scenario integrates heterogeneous sensing and data processing with state-of-the art virtual human technology, and psychology and cognitive science grounded control models. The position of the user is sensed with an overhead tracking camera, and computed using the tracking software AnTS [3]. During the simulation, an m+m plugin for MathWorks Simulink[23] continuously reads the users' location. The hybrid discrete-continuous control system is implemented using MathWorks Simulink and Stateflow[24], and controls the behavior of the 3D character, by sending Behaviour Markup Language (BML) [10] commands to the character animation system SmartBody [23] via the m+m middleware. The interface between m+m and the SmartBody system is realized via a bi-directional adaptor to the ActiveMQ middleware. To accommodate for the high resource needs of components such as the tracking system, and the SmartBody 3D rendering, the system is distributed over three PCs running the Windows operating system. In the future, additional inputs to the system are planned to be integrated such as gesture recognition based on data from a custom-built "data glove", and posture as classified based on information from the Microsoft Kinect sensor[25].

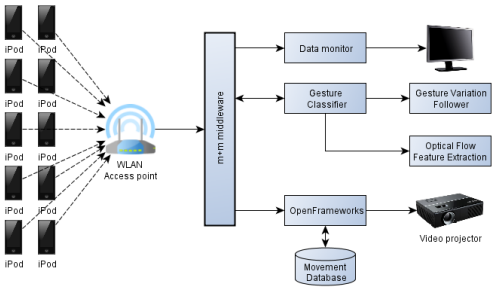


Figure 6 Architecture of the "Longing and Forgetting" multi-user interactive video installation.

Multi-user interactive video installation "Longing and Forgetting"

The "Longing and Forgetting" installation, deployed at the Surrey UrbanScreen venue in British Columbia, Canada, demonstrates the usage of the m+m middleware in constructing an interactive system involving 10 mobile devices and a central server application. The server application models and renders intelligent video agents that respond to user input from the mobile applications, and the result is projected onto an outdoor screen (Figure 6).

In the installation, participants use mobile devices (Apple iPod touch) to select and control agents that are projected onto a wall by pointing the devices at the agents and then moving in the desired direction. The accelerometer and gyroscope sensor data is filtered and combined on the mobile device, and sent to the server via the middleware. The server then performs further processing on the input data to determine the selection of a virtual agent (done via hovering over an agent) and then movement (fast jerking motion of the pointer along a certain direction). Once a movement command is issued to an agent, the internal transition of the agent is computed and an output video is selected from the movement database to execute the movement. In this example, the middleware facilitates the discovery, connection and communication of sensor data between the mobile devices and the server. Sensor processing can be done both on the mobile device, as well as the server application, depending on the computational requirements and desired features. Software bindings for the middleware interface are implemented for both the mobile and desktop platforms, and can be used by any application on supported platforms.

[23] http://mathworks.com/products/simulink/
[24] http://mathworks.com/products/stateflow
[25] http://microsoft.com/en-us/kinectforwindows/

Another feature of the system afforded by the middleware is that the sensor data, if desired for testing, deployment of new features, etc., can be dynamically plugged into other systems on the network, without any modifications either to the code or the operational mode of the application running on the existing devices.

SUMMARY AND CONCLUSION

In this paper we present the m+m middleware, the development of which is motivated by the unprecedented confluence of trends in embodied cognition, affective computing, and quantified self with a surge in the proliferation of affordable sensing devices. With its unique combination of ease of setup and configuration, high performance, and flexibility, m+m facilitates the development and deployment of distributed, real-time interactive systems in artistic, research, and commercial domains. Current limitations of m+m are that not all operations can be done via the graphical user interface, and the lack of a generic data visualization module. Future steps in the development of m+m include built-in support for generic multisensory data fusion and cross-modal mapping, a tighter integration of feature extraction methods, and the addition of further capabilities to the graphical user interface.

ACKNOWLEDGMENTS

This work was partially supported by "Moving Stories" and "m+m: Moving and Meaning" Canadian SSHRC and CANARIE grants respectively.

REFERENCES

1. Omid Alemi, Philippe Pasquier, and Chris Shaw. 2014. Mova: Interactive Movement Analytics Platform. *Proceedings of the 2014 International Workshop on Movement and Computing - MOCO '14*, ACM Press, 37–42. http://doi.org/10.1145/2617995.2618002
2. Pierre-Alain Avouac, Philippe Lalanda, and Laurence Nigay. 2012. Autonomic management of multimodal interaction. *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '12*, ACM Press, 35. http://doi.org/10.1145/2305484.2305493
3. Sergi Bermúdez i Badia, Ulysses Bernardet, Mario Negrello, Markus Knaden, and Paul F.M.J. Verschure. 2005. AnTS: A 3-Dimensional Tracking System for Behavioral Analysis of Flying Insects and Robots. http://doi.org/10.13140/RG.2.1.5008.0806
4. Ulysses Bernardet and Paul F M J Verschure. 2010. iqr: A Tool for the Construction of Multi-level Simulations of Brain and Behaviour. *Neuroinformatics* 8, 2: 113–34. http://doi.org/10.1007/s12021-010-9069-7
5. Philip A. Bernstein. 1996. Middleware: a model for distributed system services. *Communications of the ACM* 39, 2: 86–98. http://doi.org/10.1145/230798.230809
6. Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, and Nicolas Rasamimanana. 2009. Continuous Realtime Gesture Following and Recognition. In *Lecture Notes in Computer Science*, Stefan Kopp and Ipke Wachsmuth (eds.). Springer, Berlin, Heidelberg, 73–84. http://doi.org/10.1007/978-3-642-12553-9_7
7. Chris Branton, Brygg Ullmer, Andre Wiggins, et al. 2013. Toward rapid and iterative development of tangible, collaborative, distributed user interfaces. *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems - EICS '13*, JUNE: 239. http://doi.org/10.1145/2494603.2480312
8. Steve DiPaola and Caitlin Akai. 2006. Designing an adaptive multimedia interactive to support shared learning experiences. *ACM SIGGRAPH 2006 Educators program on - SIGGRAPH '06*: 14. http://doi.org/10.1145/1179295.1179310
9. Brad Johanson, Armande Fox, and Terry Winograd. 2002. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing 1*, 67–74. http://doi.org/10.1109/MPRV.2002.1012339
10. Stefan Kopp, Brigitte Krenn, Stacy Marsella, et al. Towards a Common Framework for Multimodal Generation : The Behavior Markup Language. 205–217.
11. R. Laban and F. C. Lawrence. 1974. *Effort: Economy of Human Movement*. Macdonald and Evans.
12. R. Laban and L. Ullmann. 1971. *Mastery of Movement*. Macdonald & Evans Ltd.
13. Asa Macwilliams, Christian Sandor, Martin Wagner, Martin Bauer, Gudrun Klinker, and Bernd Bruegge. 2003. Herding Sheep: Live System Development for Distributed Augmented Reality. *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, 123 – 132.
14. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. *Proceedings of the 24th Annual Symposium on User Interface Software and Technology (UIST'11)*, 315–325. http://doi.org/10.1145/1979742.1979691
15. Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 2006. YARP: Yet Another Robot Platform. *International Journal of Advanced Robotic Systems*, 1. http://doi.org/10.5772/5761
16. R. W. Picard. 2003. Affective computing: challenges. *International Journal of Human-Computer Studies* 59, 1-2: 55–64.
17. Hennadiy Pinus. 2004. Middleware: Past and present a comparison. 1–5. Retrieved February 6, 2014 from http://userpages.umbc.edu/~dgorin1/451/middleware/middleware.pdf
18. Miller Puckette. 1996. Pure Data: another integrated computer music environment. *International Computer Music Conference*, 37–41. Retrieved February 22, 2014 from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.3903
19. Daniel Pustka, Manuel Huber, Christian Waechter, et al. 2011. Automatic configuration of pervasive sensor

networks for augmented reality. *IEEE Pervasive Computing* 10, 3: 68–79. http://doi.org/10.1109/MPRV.2010.50

20. Casey Reas and Ben Fry. 2007. *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press.
21. Maryam Saberi, Ulysses Bernardet, and Steve DiPaola. 2014. An Architecture for Personality-based, Nonverbal Behavior in Affective Virtual Humanoid Character. *Procedia Computer Science* 41: 204–211. http://doi.org/10.1016/j.procs.2014.11.104
22. Etienne Schneider and F Picioroagă. 2004. Dynamic reconfiguration through OSA+, a real-time middleware. *International Middleware Doctoral Symposium*: 319–323. http://doi.org/10.1145/1028480.1030196
23. Ari Shapiro. 2011. Building a character animation system. *Motion in Games*, 98–109. Retrieved from http://www.springerlink.com/index/L24P125448583571.pdf
24. Emily Singer. 2011. The Measured Life. *MIT Technology Review* July/Augus. Retrieved March 11, 2015 from http://www.technologyreview.com/featuredstory/424390/the-measured-life/
25. Dean Takahashi. 2015. The top 11 tech trends of the Consumer Electronics Show. Retrieved March 11, 2015 from http://venturebeat.com/2015/01/12/the-top-11-tech-trends-of-the-consumer-electronics-show/
26. Andrew D Wilson and Sabrina Golonka. 2013. Embodied Cognition is Not What you Think it is. *Frontiers in psychology* 4, February: 58. http://doi.org/10.3389/fpsyg.2013.00058